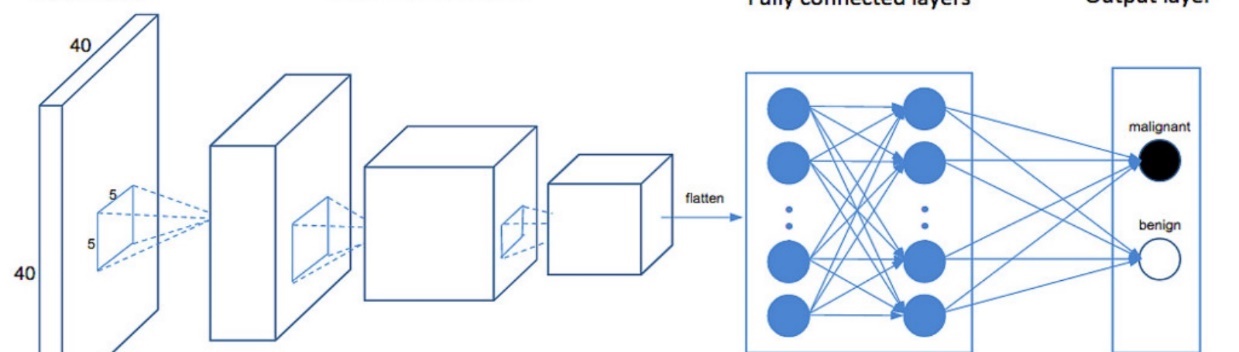
**פרויקט מסכם במדע חישובי כיתה י"ב**

מודלי למידת מכונה לאבחנה בין גידולים ממאירים ושפירים בשד

Machine Learning Models for

Breast Cancer Data

[[1]](#footnote-1)

**מגישים:**

**דניאל פלד ורועי דביר**

**מורה:**

**שלמה רוזנפלד**

**יוני 2019**

# תקציר:

סרטן השד (Breast Cancer) הוא אחד מסוגי הסרטן הנפוצים בקרב נשים. אבחון מוקדם של סרטן השד (גידולים ממאירים) מאפשר טיפול קליני בזמן לנשים שחולות בו, בטרם הסרטן התפשט ושלח גרורות ומגדיל את הסיכויים להחלמה.

מטרת עבודה זאת הינה לנבא ברמת דיוק גבוהה, בהינתן סט של תכונות ((features שהתקבלו מתוך נתוני סריקה דיגיטלית של רקמת שד של נשים, האם האישה הנבדקת בריאה או נושאת גידול סרטני ממאיר בשד, וזאת באמצעות יישום של מודלים שונים של למידת מכונה.

בדקנו ארבעה מודלים קלאסיים של למידת מכונה שהינם:

1. Logistic Regression
2. K-Nearest Neighbors
3. Decision Tree Classifier
4. Random Forest Classifier

מכל המודלים הקלאסיים, במודל ה- Logistic Regression קיבלנו את רמת דיוק הגבוהה ביותר בכל הפרמטרים, וזאת כאשר השתמשנו לאימון המודל רק ב-5 התכונות הטובות ביותר לניבוי, תוך השמטת יתר התכונות. בממוצע 98% מהניבויים על נתוני המבחן היו נכונים, 96% מהנשים הנושאות גידול ממאיר נובאו נכון ככאלה הנושאות גידול ממאיר (98% מהנשים הבריאות נובאו כבריאות. במודל ה-K-Nearest Neighbors, כאשר השתמשנו ב-10 התכונות הטובות ביותר לניבוי, התקבלו אחוזי דיוק דומים, כאשר 95% מהנשים הנושאות גידול ממאיר נובאו נכון כנושאות גידול ממאיר ו-99% מהנשים הבריאות נובאו כבריאות. רמת הדיוק במודל ה- Decision Tree Classifier, הייתה נמוכה בהרבה ובלתי מספקת. שיפור גדול הושג במעבר ממודל זה למודל ה Random Forest Classifier, אם כי רמת הדיוק נמוכה במקצת ביחס למודלים הראשונים.

בנינו גם מודל מורכב יותר, של למידה עמוקה. הצלחנו להגיע לרמות דיוק גבוהות מאוד עד לכדי 100% דיוק מבחינת מדדי ההערכה הנ"ל. תוצאות אלו אף טובות יותר מאלו שהושגו באמצעות שימוש באלגוריתמי למידת מכונה קלאסיים.

# תוכן עניינים:

[מבוא 5](#_Toc11841530)

[מטרת העבודה 7](#_Toc11841531)

[הבעיה אתה מתמודדת העבודה 7](#_Toc11841532)

[שאלת החקר 7](#_Toc11841533)

[שאלות חקר משניות 7](#_Toc11841534)

[האופנים והדרכים לביצוע העבודה שנועדו לעזור לענות על שאלת החקר 7](#_Toc11841535)

[אילוצים ומגבלות 8](#_Toc11841536)

[בסיס הנתונים 9](#_Toc11841537)

[הצגת הנתונים 10](#_Toc11841538)

[הצגת הנתונים שבקובץ הנתונים באמצעות היסטוגרמות 12](#_Toc11841539)

[צמצום של התאמת יתר (overfitting) 19](#_Toc11841540)

[הכנת הנתונים וכלי לסטטיסטיקה וניתוח (נמצאים גם בקבצי הקוד) 21](#_Toc11841541)

[הכנת הנתונים לבדיקתם – במודלי למידת מכונה (גרסה ראשונה) 21](#_Toc11841542)

[הכנת פונקציה המחשבת ומשרטטת "מטריצת הטעויות" ((confusion\_matrix – במודלי למידת מכונה (גרסה ראשונה) 22](#_Toc11841543)

[הכנת הנתונים לבדיקתם – במודלי למידה עמוקה (גרסה שנייה- מעודכן ומותאם) 23](#_Toc11841544)

[הכנת פונקציה המחשבת ומשרטטת "מטריצת הטעויות" ((confusion\_matrix – במודלי למידה עמוקה (גרסה שנייה- מעודכן ומותאם) 23](#_Toc11841545)

[למידת מכונה 24](#_Toc11841546)

[מודל Logistic Regression 24](#_Toc11841547)

[K-Nearest Neighbors model (KNN) 30](#_Toc11841548)

[Decision Tree Classifier model (DT) 33](#_Toc11841549)

[Random Forest Classifier model (RF) 36](#_Toc11841550)

[למידה עמוקה 38](#_Toc11841551)

[רקע תיאורטי 38](#_Toc11841552)

[Forward-Propagation 40](#_Toc11841553)

[אלגוריתם Backpropagation 40](#_Toc11841554)

[תהליך האופטימיזציה – Gradient Descent 44](#_Toc11841555)

[מבנה התוכנה והאלגוריתם החישובי (Layer , Network) 44](#_Toc11841556)

[תוצאות 47](#_Toc11841557)

[טיב הרשת כתלות במספר הנוירונים עבור שכבה נסתרת אחת 47](#_Toc11841558)

[טיב הרשת כתלות בקצב הלמידה שתי שכבות נסתרות 48](#_Toc11841559)

[טיב הרשת כתלות בגודל ה-Batch על הסט עבור שתי שכבות נסתרות 50](#_Toc11841560)

[טיב הרשת כתלות בכמות השכבות (המשתמשות בפונקציות אקטיבציה זהות) 51](#_Toc11841561)

[טיב הרשת כתלות בפונקציות האקטיבציה וסדרן על הלמידה עבור שתי שכבות נסתרות 53](#_Toc11841562)

[אמינות התוצאה והשגיאה 53](#_Toc11841563)

[סיכום הפרויקט ומסקנות 54](#_Toc11841564)

[נספחים 56](#_Toc11841565)

[נספח 1- ביבליוגרפיה (לפי הפניות בעבודה) 56](#_Toc11841566)

[נספח 2 – קישורים לכל קבצי המודלים 58](#_Toc11841567)

[נספח 3- הפניות לקוד בתוך המסמך (ההפניות מסומנות בעבודה בירוק) 59](#_Toc11841568)

**קישורים לכל קבצי ב- cocalc נמצאים בנספח 2 של הפרויקט (בעמוד האחרון של המסמך). מיקום קבצי המודלים:**

1) כל הקבצים של למידת מכונה קלאסית (5) ממוקמים בתיקיית note של התלמיד דניאל פלד.

2) כל הקבצים של למידה עמוקה (2) ממוקמים בתיקיית Project של התלמיד רועי דביר.

# מבוא:

סרטן השד (Breast Cancer) הוא אחד מסוגי הסרטן הנפוצים בקרב נשים. אבחון מוקדם של סרטן השד יכול להגדיל את הסיכויים להחלמה ממנו באופן משמעותי וזאת מכיוון שאבחון מוקדם של סרטן השד (גידולים ממאירים) מאפשר טיפול קליני בזמן לנשים שחולות בו, בטרם הסרטן התפשט ושלח גרורות. בנוסף, חשוב לאבחן נכון בין גידולים ממאירים לגידולים שפירים. סיווג מדויק של גידולים ממאירים מאפשר טיפול קליני ומגדיל את סיכויי החלמה. חשוב גם לסווג גידולים שפירים נכון. סיווג שגוי של גידול שפיר כממאיר יגרום למתן טיפולים לנשים בריאות, הכרוכים בסבל פיזי ונפשי מיותר . מהנימוקים הנ"ל, מדובר בבעיית סיווג ("קלסיפיקציה") עם חשיבות רפואית רבה.

מטרת עבודה זאת הינה לנבא ברמת דיוק גבוהה, בהינתן סט של תכונות ((features שהתקבלו מתוך נתוני סריקה דיגיטלית של רקמת שד של נשים, האם האישה הנבדקת בריאה או נושאת גידול סרטני ממאיר בשד, וזאת באמצעות יישום של מודלים שונים של למידת מכונה. בסיס הנתונים בו השתמשנו נקרא Breast Cancer Wisconsin (Diagnostic) Data Set [[2]](#footnote-2) והוא מכיל מאגר של 569 נשים, שלכל אחת מהן נערכה סריקה דיגיטלית ונמדדו ערכים של 30 תכונות שונות.

בינה מלאכותית (Artificial Intelligence- AI ) הינה תחום במדעי המחשב שעניינה עיסוק בתכנות מחשבים באופן שיציגו יכולות המאפיינות את הבינה האנושית. בינה מלאכותית נחלקת בעקרון לשתי רמות של בינה: בינה דמוית אנוש (בה המחשב חושב ומסיק מסקנות בדומה לצורת החשיבה האנושית) ולבינה שאינה דמוית אנוש (בה המחשב מגיע ליכולות עצמאיות ולבינה, ללא קשר לאופן בו פועל המוח האנושי). תחום זה הינו תחום-על במדעי המחשב והוא מאוד רחב ומכיל בתוכו תת-תחומים רבים.

למידת מכונה Machine Learning)) הינה מתודולוגיה נפוצה לפתרון בעיות קלסיפיקציה מסוג זה. למידת מכונה הינה תת תחום אחד במסגרת תחום רחב שמתפתח כיום בקצב מהיר והוא תחום הבינה המלאכותית.

למידת מכונה, שכאמור הינה תת תחום בנושא הבינה המלאכותית, היא התחום בו התמקדנו בעבודה זאת והיא תחום בו נעשה שימוש נפוץ במדע חישובי. המדובר במתודולוגיה נפוצה לפתירת בעיות קלסיפיקציה תוך שימוש במאגרי מידע גדולים. למידת מכונה, חופפת עם תחום האופטימיזציה והסטטיסטיקה ועניינה פיתוח אלגוריתמים שיעזרו למחשב ללמוד (לפתח בינה מלאכותית), באמצעות דוגמאות. מטרת תת תחום זה, הינה טיפול בנתונים ממשיים מהעולם האמתי כאשר האדם לא יכול להתמודד איתם שלבדו וגם שלא ניתן לכתוב עבורם כללי זיהוי ספציפיים ועל כן לא ניתן לפתח תוכנת מחשב שתפתור בעיות אלו. הנ"ל יכול להתרחש כאשר לא ניתן לכתוב את הכללים לקלסיפיקציה בצורה מפורשת או כשהם משתנים כתלות בגורם אחר (למשל כתלות בזמן) ואופי שינוי זה לא ניתן לתיאור בצורה מתמטית פשוטה. מטרת תת-תחום זה ככלל היא גילוי, חיזוי ומידול[[3]](#footnote-3) של עובדות לגבי העולם האמיתי. תת תחום זה מכיל בתוכו תת תחומים רבים אחרים וניתן לממש עבור אותו מאגר מידע מידול שונה, וזאת עשינו בעבודה זאת. ביצענו למידת מכונה באמצעות שימוש במודלים "פשוטים" של למידת מכונה ובאמצעות רשת נוירונים (רשת עצבים מלאכותית).

רשת נוירונים היא סוג של מודל מתמטי חישובי שנעשה בהשראת תהליכים מוחיים וקוגניטיביים המתרחשים במוח כדי לפתח מודל שישמש ללמידת מכונה. רשת שכזאת, מכילה לרוב מספר רב של יחידות מידע (קלטים ופלטים) שמקושרים זה לזה ולפעמים אף דרך שכבות חבויות (Hidden Layers). מדובר במודל נפוץ מאוד בלמידת מכונה ושימושיו רבים: ניתוח וחיזוי בשוק ההון, זיהוי כתב יד, זיהוי פנים ותמונות, זיהוי ווקאלי (קולי) וכו' . על כן, במסגרת פרויקט זה ביצענו שימוש במודל נפוץ זה על מנת לחזות (באמצעות רשת נוירונים בעלת שכבה נסתרת עם קלטים רבים האם אישה מסוימת, על סמך התכונות המאפיינות את דגימת רקמת תאי השד, חולה בסרטן השד או שאינה חולה (בהמשך המסמך רקע תאורטי מפורט ורחב יותר שמסביר על האלגוריתם של המודל שבנינו).

בעבודה זו בדקנו ארבעה מודלים פשוטים של למידת מכונה ומודל מורכב יותר של למידה עמוקה. המודלים נכתבו בשפת Python בסביבת העבודה cocalc[[4]](#footnote-4). מודלי למידת המכונה מומשו באמצעות חבילת scikit-learn ומודל הלמידה העמוקה (השני שיצרנו) מומש באופן עצמאי תוך שאיבת השראה מחבילת keras. למעשה, את המודל הגנרי הראשון שלנו שבדקנו (שנזכיר כעת שהוא קיים, אך לא נכתוב עליו בעבודה כי הוא מבוסס על עקרונות דומים כמו של המודל הגנרי השני) והפכנו לאופטימלי, יצרנו בעזרת keras כדי להראות את האופן הכללי של ביצוע הלמידה של רשת נוירונים (המודל של keras מוסבר בקוד). ההתרכזות העיקרית שלנו במסמך היא החקר של הנושא וההבנה איך הכל מחובר יחדיו מבחינות שונות.

# מטרת העבודה

בעבודה זו ננסה לחזות ברמת דיוק גבוהה, בהינתן בסיס נתונים המכיל תכונות מסוימות (features) שהתקבלו מניתוח סריקות דיגיטליות של תאים שנלקחו מדגימות ביופסיה מרקמה משד של נשים, האם הגידול בשד הינו שפיר או ממאיר.

## הבעיה אתה מתמודדת העבודה

סרטן השד הוא הסרטן בו אחוזי התמותה לנשים הוא גבוה מאוד (17%). מדובר בסרטן שכיח מאוד. לדוגמא, 100,000 נשים בישראל חלו בסרטן זה בשנה החולפת. מתוכן, כ17,000 נפטרו. גילוי מוקדם של סרטן זה והתחלת טיפול בשלב מוקדם מפחית בצורה ניכרת את הסיכוי לתמותה, ושומר על איכות החיים של המטופלות. עבודה זאת, באמצעות דיון במודלים שונים, באה לדון במודלים שונים הבודקים האם לנבדקת יש סרטן שד ממאיר או שמא היא בריאה. אנו רוצים למצוא מודלים המבוססים על למידת מכונה שיאפשרו לשפר את איכות חיי הנשים ולהאריך את חייהן ע"י אבחון מוקדם ומדויק של סרטן השד.

## שאלת החקר:

בהינתן סט של תכונות שמקורן בדגימת תאים שנלקחה מרקמת שד של אישה, האם ניתן לחזות ברמת דיוק גבוהה, האם האישה נושאת גידול ממאיר או שהיא בריאה?

## שאלות חקר משניות:

כיצד כל מודל מתאר את המציאות (באיזה דיוק)? כיצד ניתן לשפר את הדיוק של כל אחד מהמודלים?

אילו מודלים מתארים בדיוק מרבי את המציאות?

כיצד שינוי של "קו הגבול" לקלסיפיקציה במודלים משפיעה על אבחון מדויק ונכון של הנבדקות?

## האופנים והדרכים לביצוע העבודה שנועדו לעזור לענות על שאלת החקר:

בסיס הנתונים בו השתמשנו נקרא Breast Cancer Wisconsin (Diagnostic) Data Set [[5]](#footnote-5) . בסיס הנתונים מכיל מאגר של 569 נשים, שלכל אחת מהן ערכים של 30 תכונות שונות. בחלק מבסיס הנתונים נשתמש על מנת לאמן (train) את המודל ובחלק מבסיס הנתונים נשתמש כנתוני מבחן (test) על מנת לבדוק את דיוקו. נסביר כיצד נבנה בסיס הנתונים ועל ידי מי, אילו נתונים הוא מכיל ומה משקפים הנתונים.

נציג את בסיס הנתונים, ננתח אותו ניתוח ראשוני, ועל בסיס הניתוח הראשוני נעלה השערות בדבר התכונות (features) המשפיעות ביותר על הקלסיפיקציה. נשאף לצמצם את מספר התכונות המשמשות לאימון המודל, על מנת להימנע מהתאמת יתר לנתוני המבחן.

בהמשך ייבחנו מספר מודלים פשוטים של למידת מכונה, על בסיס הנתונים, תוך שימוש בחבילת sklearn שבפייתון:logistic regression, K-Nearest Neighbors, Decision Tree, Random Forest .נציג את הרקע התאורטי של כל אחד מהמודלים, יתרונותיו וחסרונותיו.

נשווה את הדיוק של כ"א מהמודלים בחיזוי על נתוני המבחן, לאחר אימון המודל, נבחן את ה- cross validation של כל מודל ונבדוק לגבי כל מודל מהן התכונות החשובות ביותר, בהן ניתן להשתמש לאימון המודל, תוך ויתור על התכונות החשובות פחות באימון המודל מצד אחד והגדלת דיוק המודל מהצד השני .

בהמשך, נראה שבנינו מודל מורכב יותר של "למידה עמוקה" (deep learning), באמצעות רשת נוירונים מלאכותית. נציג את הרקע התאורטי של המודל וננתח את תוצאותיו בהשוואה למודלים הפשוטים יותר.

בסופו של דבר, נסכם את הפרויקט ונסיק מסקנות כלליות לעבודה.

## אילוצים ומגבלות

לצערנו, מאגרי המידע בתחום מוגבלים. מצאנו מאגרים המכילים יותר נשים מאשר במאגר שלנו, עם פחות תכונות או עם תכונות אחרות. בחרנו במאגר לא גדול במיוחד המכיל 569 נשים מאזור מסוים – ויסקונסין בארה"ב. המאגר אינו "מאוזן" הואיל והוא מכיל יותר נשים בריאות מאשר נשים חולות -כ-63% נשים בריאות ו-37% נשים חולות, דבר שעלול לגרום להתאמת יתר של המודלים למאפיינים של הנשים הבריאות. מאידך, אם "נאזן" את המאגר על ידי שימוש חלקי בנתוני הנשים הבריאות, תוך השמטת חלק מנתוני הנשים הבריאות, נאבד אחוז משמעותי מהיקף המאגר ונישאר עם מאגר אף מצומצם יותר.

יש להניח כי לרקע אתני ולמאפיינים הגנטיים יש השפעה על התכונות. המאגר שלנו משקף אוכלוסייה שמידת ההטרוגניות שלה בלתי ידועה, שהמכנה המשותף שלה היא מגורים באזור ויסקונסין, ארה"ב. המגבלות הללו עלולות לגרום לתוצאות המודלים להיות פחות מדויקות.

# בסיס הנתונים:

בסיס הנתונים נקרא: [[6]](#footnote-6)Breast Cancer Wisconsin (Diagnostic) Data Set  . בסיס הנתונים נוצר על ידי ד"ר וויליאם וולברג, רופא בבית החולים של אוניברסיטת ויסקונסין במדיסון, ויסקונסין, ארה"ב. ד"ר וולברג לקח דגימות ביופסיה של רקמת תאי שד מנשים בריאות ומנשים חולות בסרטן השד. הוא ביצע סריקה דיגיטלית של הדגימות, עשה שימש בתוכנת מחשב גרפית, ששמה Xcyt, על מנת לבצע ניתוח של תכונות ציטולוגיות (תכונות של תאים בדגימה) המבוססות על הסריקה הדיגיטלית. תכנת המחשב הנ"ל מחשבת מתוך הסריקות הדיגיטליות את עשר התכונות (features) של כל אחד מהתאים בדגימה ולאחר מכן את הערך הממוצע (mean value), את סטיית התקן (Standard Deviation) ואת ערך הקיצון (Worse Value) של כל תכונה בכל התאים שבדגימה. כך מתקבל, לגבי כל דגימה, וקטור המורכב מ-30 ערכים – הערך הממוצע, סטיית התקן וערך הקיצון של כל אחת מעשר התכונות הנבדקות. קובץ הנתונים, שהורדנו מהאתר שפרטיו בהערת שוליים מס' 2, נשמר על ידינו כקובץ אקסל. עשר התכונות (features) הן:

1. רדיוס התא (radius).
2. טקסטורה (texture)- ככל והערך עולה, התא פחות "חלק" ומכיל יותר ריקמה גושית. ערך הטקסטורה מחושבת לפי ה- grey level סביב הפיקסלים של צילומי התאים.
3. היקף התא ((perimeter.
4. שטח התא (area).
5. אחידות/ "חלקות" התא – הנמדדת לפי מידת הסטייה המקומית בתוך התא של אורך הרדיוסsmoothness) ).
6. קומפקטיות התא (compactness)- היחס בין היקף התא בריבוע לבין שטח התא פחות 1. הקומפקטיות נמדדת לפי היחס: perimeter² / (area-1 ). ככל והתא סוטה מצורה של עיגול, הקומפקטיות גדלה.
7. קעירות התא ((concavity- דרגת הקעירות של מתאר התא - Severity of concave portions of the contour
8. מספר נקודות קעורות – מספר החלקים הקעורים בקווי המתאר (concave points).
9. מידת הסימטריה של התא (symmetry).
10. ממד פרקטלי (fractal dimension)- מודד את "מורכבות" התא לפי היחס בין השינוי ברמת הפרטים לבין השינוי בסקלת המדידה ("קו החוף" – coastline של התא).

## הצגת הנתונים**[[7]](#footnote-7)**:

ייבוא ספריות פייתון לצורך הצגת הנתונים ( הקוד נמצא בנספח 3.1)

הקובץ מכיל 569 דגימות מנשים שונות, ייבאנו את הקובץ כקובץ pandas:

#importing the dataset

dataset = pd.read\_csv('/home/user/data/Breast\_Cancer\_Data\_CSV.csv')

#Making a dataset by reading the data with panda, returned as two-dimensional data structure/data frame with labeled axes

dataset.drop(['ID number'],axis=1,inplace=True)

#Dropping the feature ID since it has nothing to do with predicting cancer

dataset.head()

print("Cancer data set dimensions : {}".format(dataset.shape))

#Printing shape size of dataset

גודל בסיס הנתונים:

Cancer data set dimensions : (569, 31)

הקובץ מכיל 569 נבדקות.

"ניפינו" מבסיס הנתונים את מספרי תעודת הזהות של הנבדקות, משום שמספר תעודת הזהות כמובן שאינו מהווה feature שישמש אותנו לקלסיפיקציה. לשם כך השתמשנו בפקודה:

dataset.drop(['ID number'],axis=1,inplace=True)

להלן עשרת הטורים הראשונים של הטבלה עם חמש השורות הראשונות של הטבלה:



בדקנו האם חסרים נתונים כלשהם בקובץ בעזרת הפקודה:

dataset.isnull().sum()

קיבלנו בהרצה שלא חסרים נתונים כלשהם לגבי אף תכונה אצל נבדקת כלשהיא. מאחר ולא חסרים נתונים כלשהם, עשינו שימוש בכל קובץ הנתונים. אם היינו מוצאים שלגבי מספר מצומצם של נבדקות חסרים נתונים, היינו מנפים את אותן נבדקות מקובץ הנתונים.

הדיאגנוזה מסומנת על ידי האותיות "M" ו-"B". החלפנו את "M" במספר 1 (מסמן גידול ממאיר) ואת "B" במספר 0 (שפיר).

dataset['Diagnosis'] = dataset['Diagnosis'].map({'M':1,'B':0})

בעזרת הפקודה:

dataset.Diagnosis.value\_counts()

בדקנו כמה מתוך 569 הנשים הנבדקות נמצאו עם גידול ממאיר וכמה נמצאו עם גידול שפיר. 357 נשים נמצאו עם אבחנה של "שפיר", 212 נשים נמצאו עם אבחנה של "ממאיר". העובדה שמספר הנתונים על נשים עם "שפיר" גדול יותר ממספר הנתונים שח נשים עם "גידול ממאיר" עלולה לגרום "התאמת יתר" באימון המודל, לכיוון זיהוי של נשים עם "גידול שפיר". מאחר ובסיס הנתונים אינו גדול יחסית, החלטנו לא לצמצם את קובץ הנתונים לגבי נשים עם "גידול שפיר" כך שכמות הנתונים

## הצגת הנתונים שבקובץ הנתונים באמצעות היסטוגרמות:

בחירת ה- features להצגה בהיסטוגרמות:

#select the features to be vizualized

features=list(dataset.columns[0:10])

בדוגמת הרצה זו, נבחרו 10 התכונות (הטורים) הראשונים, המשקפים את הממוצע של התכונות הנבדקות.

חלוקת בסיס הנתונים לשתי קבוצות, בהתאם לדיאגנוזה: (הקוד נמצא בנספח 3.2)

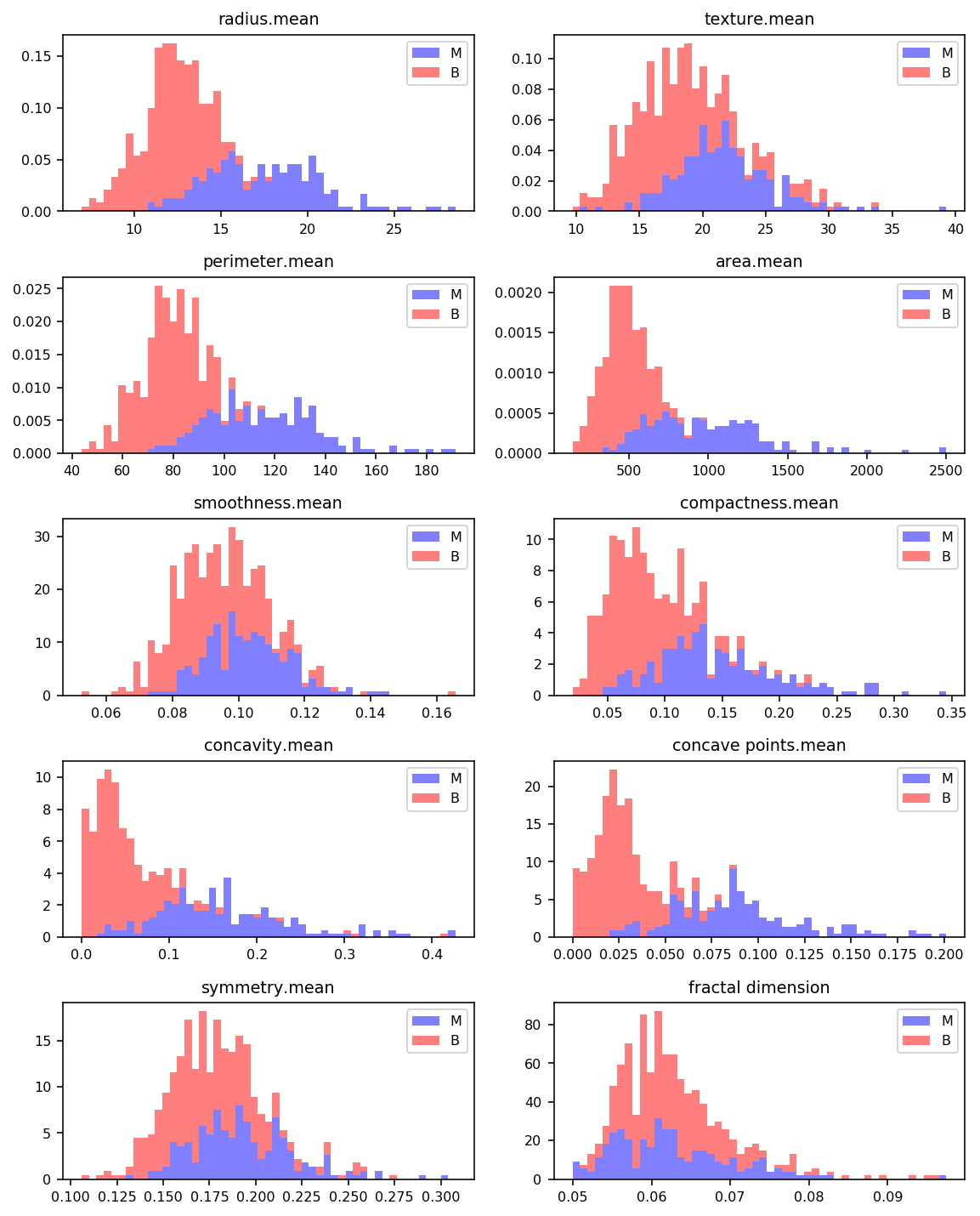
שרטוט היסטוגרמות לתכונות הנבדקות, בצבע שונה לכל אחת מהקבוצות (אדום – שפיר, כחול ממאיר): (הקוד נמצא בנספח 3.3)

תכונות "הממוצע", סטיית התקן וערכי הקיצון מוצגות באיורים שבעמודים הבאים.

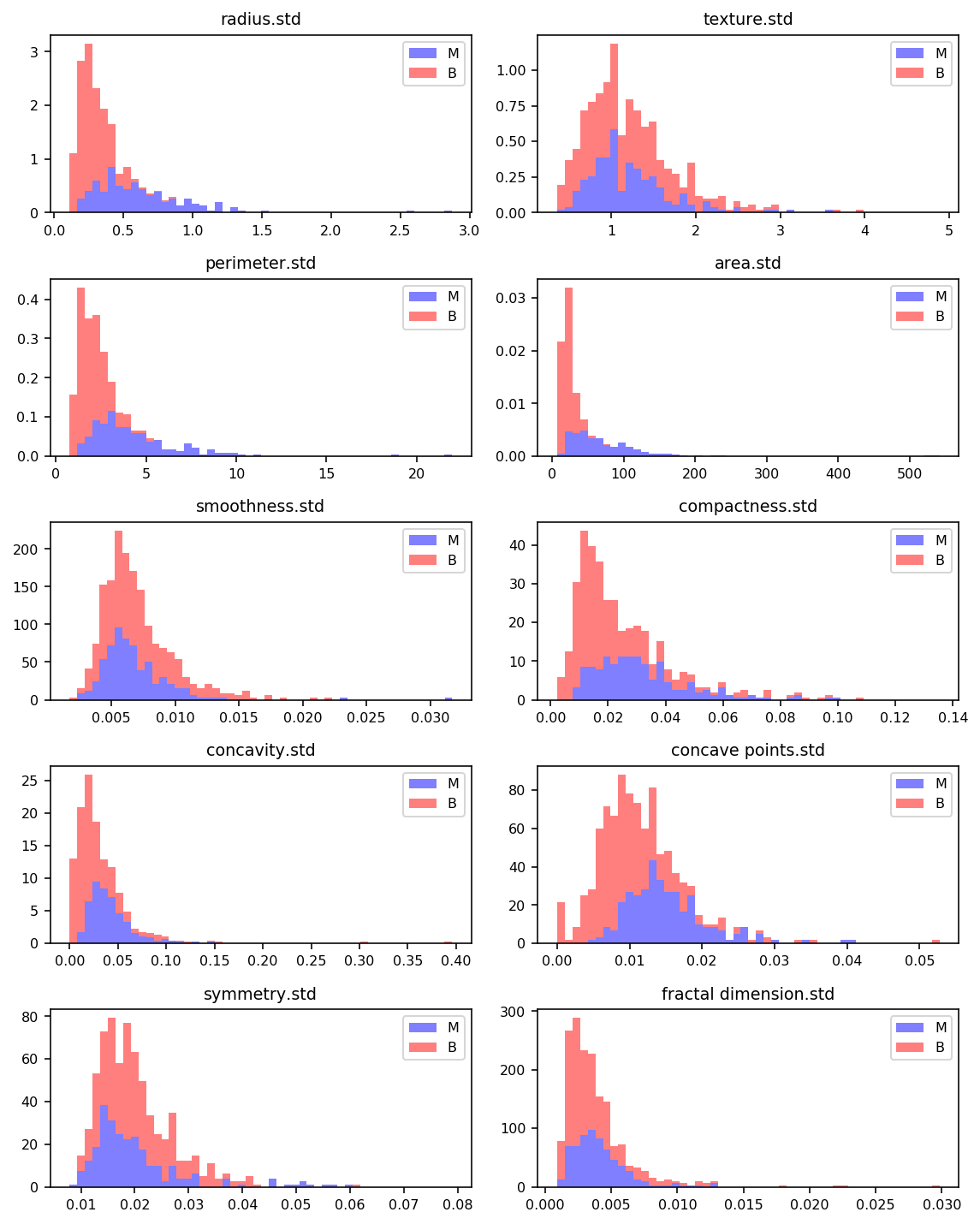
מההיסטוגרמות של הממוצעים ניתן לראות שהתכונות בהן יש הפרדה ברורה בין שתי הקטגוריות הן: **רדיוס ממוצע, היקף ממוצע, שטח ממוצע, קעירות תא ממוצעת ומספר נקודות הקעירות**: הערכים של התכונות הנ"ל גבוהים באופן מובחן אצל מרבית הנשים החולות (אם כי יש חפיפה מסוימת בין שתי הקבוצות, בתחומי החפיפה סיווג יהיה קשה יותר). לעומת זאת, בהתייחס ליתר התכונות, ההפרדה הרבה פחות ברורה ויש חפיפה רבה בין ההיסטוגרמות של שתי הקטגוריות. ניתן לשער כי הרדיוס ממוצע, ההיקף הממוצע, השטח הממוצע, קעירות התא הממוצעת ומספר נקודות הקעירות הן התכונות החשובות ביותר לסיווג.

מההיסטוגרמות עבור סטיית התקן רואים כי קיימת חפיפה רבה בין סטיות התקן של התכונות בנשים חולות ובריאות. מכאן, ניתן לשער כי תכונות "סטיית התקן" לא תשמשנה לקטגוריזציה ואולי ניתן יהיה לבצע את אימון המודל ללא שימוש בנתונים של תכונות אלה.

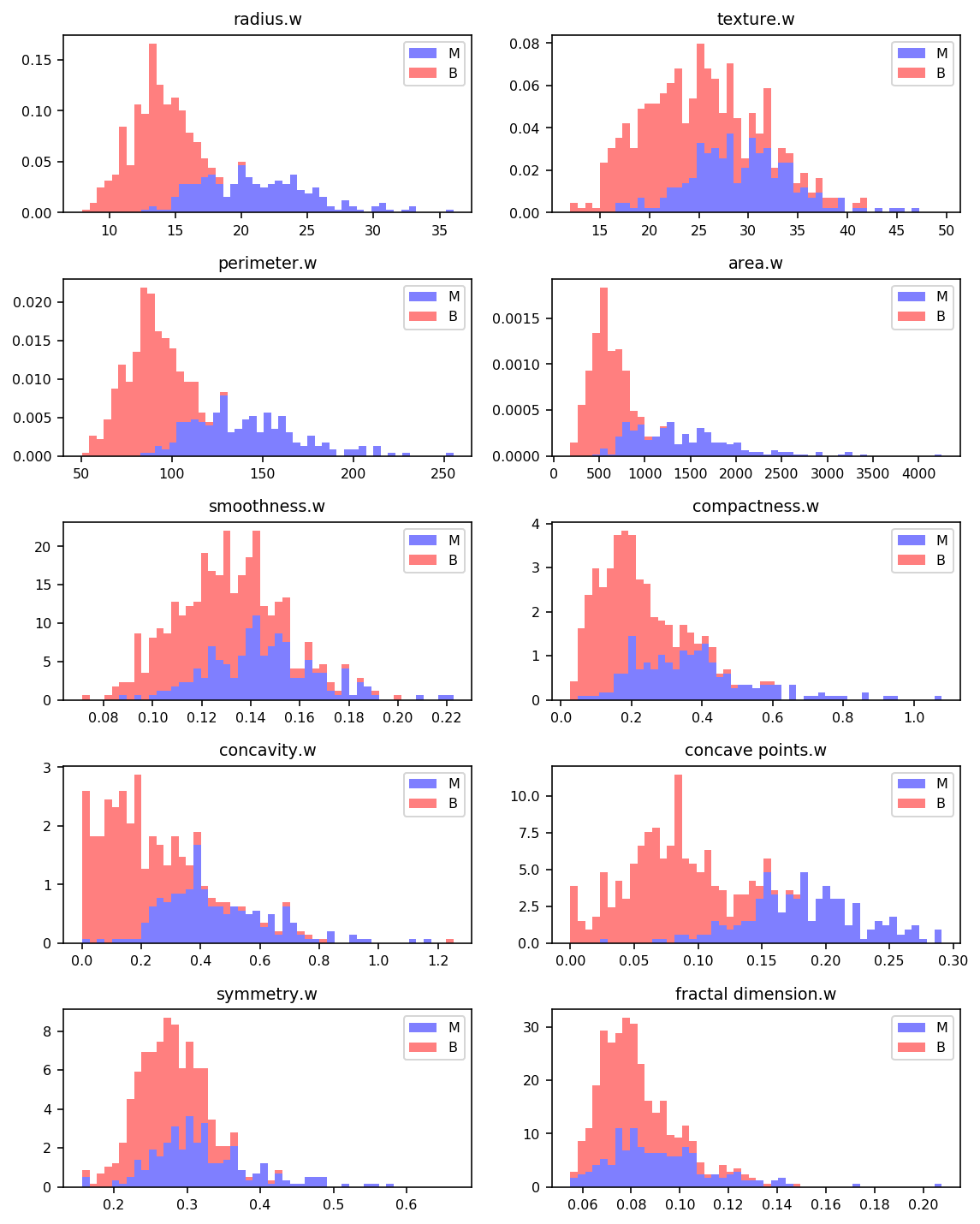
לגבי תכונות "ערכי הקיצון" – בדומה לתכונות הממוצע, ניתן לראות שהתכונות בהן יש הפרדה ברורה בין שתי הקטגוריות הן: ערך הקיצון של הרדיוס, ערך הקיצון של ההיקף, ערך הקיצון של שטח התא , ערך הקיצון של קעירות התא וערך הקיצון של מספר נקודות הקעירות בתא. ויזואלית נראה שהתכונה של ערך הקיצון של מספר נקודות הקעירות מפרידה טוב יותר מאשר התכונה של ערך הממוצע של מספר נקודות הקעירות.



איור 1. היסטוגרמות עבור ערכי הממוצע

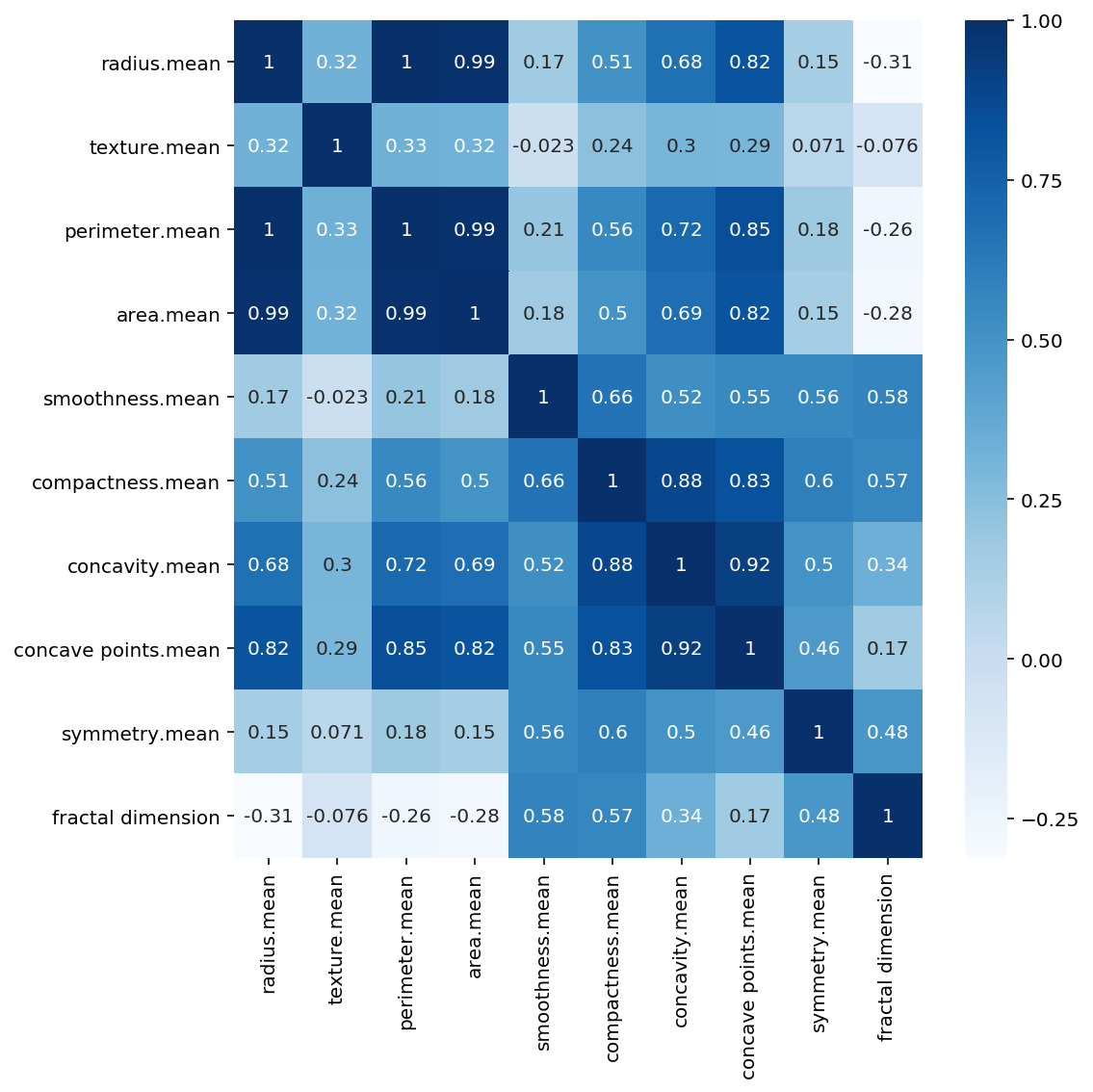


איור 2. היסטוגרמות עבור ערכי סטיית התקן



איור 3. היסטוגרמות עבור ערכי הקיצון

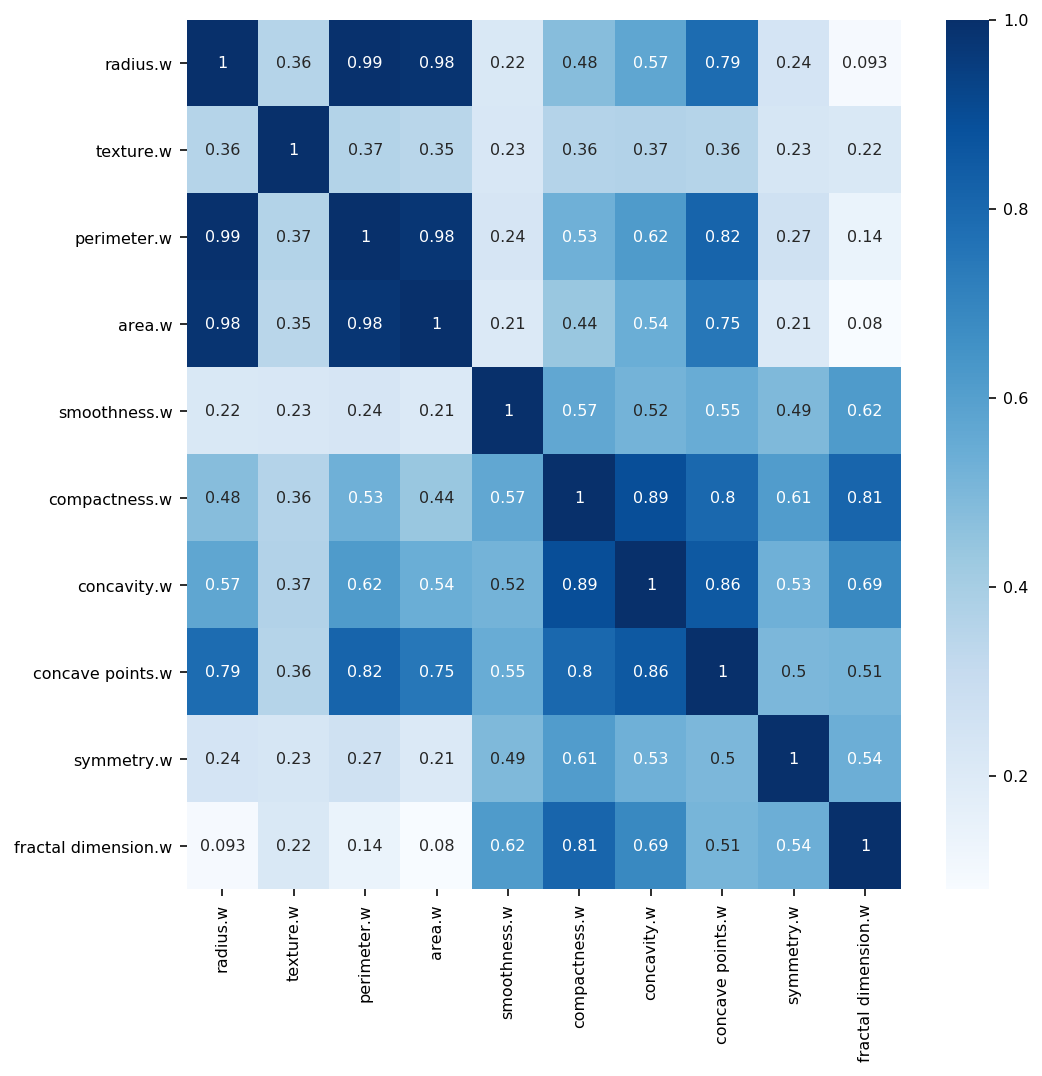
מטריצות קורלציה ביו התכונות (הקוד נמצא נספח 3.4):



איור 4 מטריצת קורלציה של תכונות הממוצע

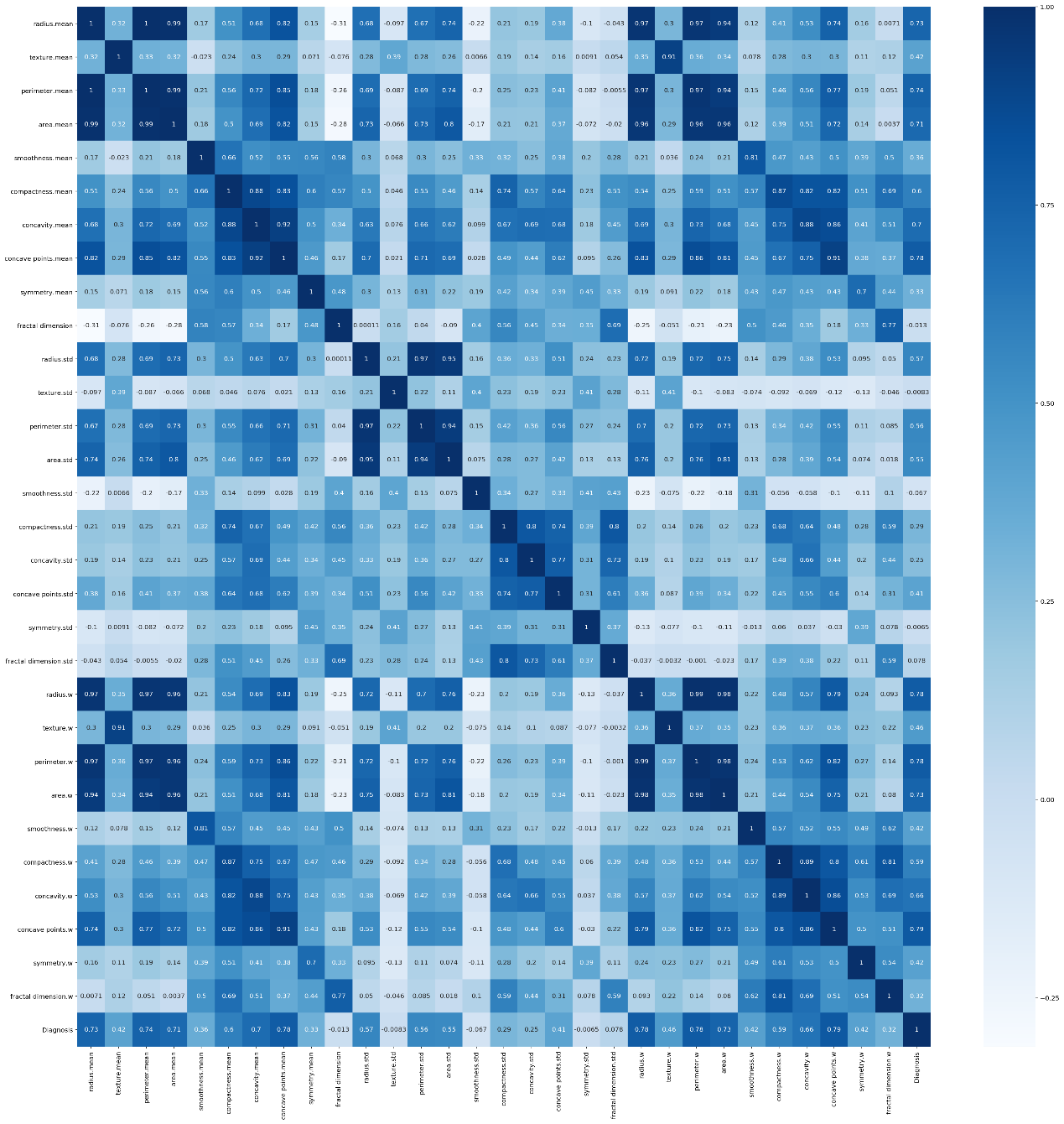
ניתן לראות כי יש קורלציה מלאה בין התכונות הבאות: רדיוס ממוצע, היקף ממוצע ושטח תא ממוצע. יש קורלציה גבוהה גם בין התכונות של קעירות התא הממוצעת ומספר נקודות הקעירות (0.92). יש קורלציה די גבוהה, אך לא מלאה- בין 0.82 ל-0.85 בין התכונה של מספר נקודות הקעירות הממוצע לבין תכונות הרדיוס/שטח/היקף. יש קורלציה נמוכה יותר, בין תכונת "הקעירות" הממוצעת לבין תכונות הרדיוס/היקף/שטח ממוצעים (בין 0.68 ל-0.72). יש קורלציה גבוהה בין התכונות הבאות: קומפקטיות ממוצעת וקעירות ממוצעת (0.88), קומפקטית ממוצעת ומספר נקודות קעירות ממוצע (0.83).

בין רוב התכונות יש קורלציה חיובית. לתכונה fractal dimension הממוצעת יש קורלציה שלילית עם תכונות הרדיוס, הטקסטורה, ההיקף והשטח הממוצעים.



איור 5. מטריצת קורלציה של תכונות ערכי הקיצון

ניתן לראות כי יש קורלציה מלאה בין התכונות הבאות: ערך רדיוס קיצון, ערך היקף קיצון ושטח תא קיצון. יש קורלציה גבוהה גם בין התכונות של ערך הקיצון של קעירות התא וערך הקיצון של מספר נקודות הקעירות (0.86). יש קורלציה די גבוהה, אך לא מלאה- בין 0.75 ל-0.82 בין מספר נקודות הקעירות לבין תכונות הרדיוס/שטח/היקף. אין קורלציה גבוהה בין ערך הקיצון של קעירות התא לבין ערכי הקיצון ש הרדיוס/היקף/שטח (בין 0.54 ל-0.67). יש קורלציה גבוהה בין התכונות הבאות: קומפקטיות –ערך קיצון וקעירות –ערך קיצון (0.89), קומפקטית ממוצעת ומספר נקודות קעירות ממוצע (0.80). באופן מקורב, מטריצות הקורלציה של תכונות "הממוצע" ותכונות "ערכי הקיצון" דומות.



איור 6. מטריצת קורלציה של כל התכונות

ניתן לראות כי קיימת קורלציה גבוהה גם בין ערכי הממוצע לערכי הקיצון- בפרט בהתייחס לרדיוס, להיקף ולשטח התא- יש קורלציה גבוהה בין ערכי הממוצע וערכי הקיצון של רדיוס, היקף ושטח התא. מניתוח ההיסטוגרמות ומטריצות הקורלציה ניתן לשער שייתכן וניתן להשתמש בווקטור תכונות מצומצם יחסית על מנת לחזות מה תהיה הדיאגנוזה.

# צמצום של התאמת יתר (overfitting)

התאמת יתר (overfitting) היא בעיה בלמידת מכונה שבה המודל מותאם יתר על המידה לאוסף מסוים של נתונים (האוסף שהיה זמין לשם אימונו) ועל כן מצליח פחות בבצוע תחזיות. התאמת יתר מתרחשת כאשר המודל נקבע על ידי יותר פרמטרים מאשר הנתונים מצדיקים. שימוש בתכונות לא רלוונטיות עלול להפחית את הדיוק של המודלים[[8]](#footnote-8).

יתרונות לצמצום מספר התכונות:

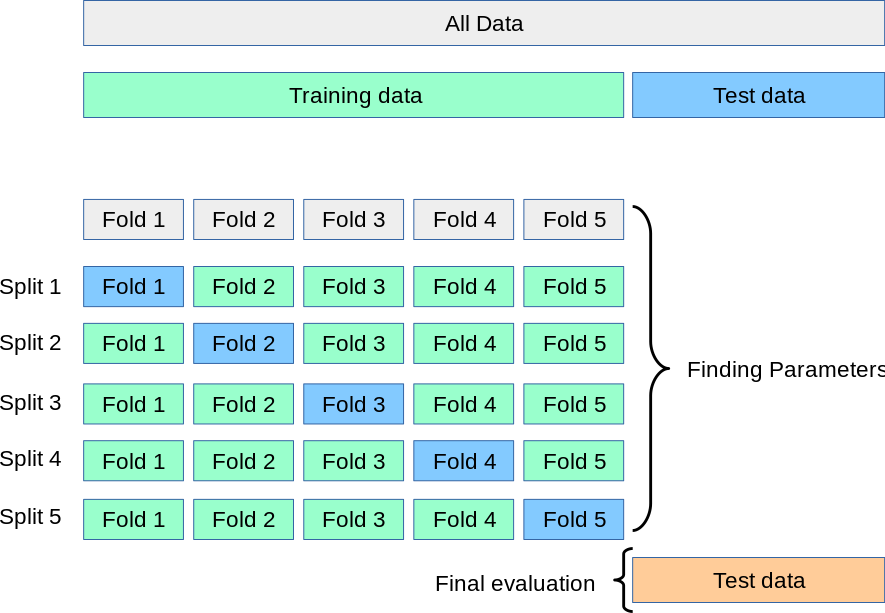
* ניפוי נתונים מיותרים מצמצם קבלת החלטות על בסיס "רעש" סטטיסטי.
* ניפוי נתונים מטעים מביא לשיפור הדיוק של המודל.
* שימוש בפחות תכונות מביא לכן האלגוריתם מתאמן מהר יותר.

עודף הפרמטרים מאפשר למודל ללמוד את הרעש הסטטיסטי כאילו הוא מייצג התנהגות אמיתית[[9]](#footnote-9).

ישנן מספר דרכים על מנת להימנע מ"התאמת יתר"[[10]](#footnote-10):

1. אימון מספר רב יותר של נתונים (אחוז גדול יותר בסיס הנתונים ישמש ל-training)- בעבודה זו לא ננקוט בדרך זו, הואיל ובסיס הנתונים אינו גדול יחסית (569 נשים בלבד). לכן עבור כל המודלים 70% מהנתונים ישמשו לאימון ((train ו-30% מהנתונים למבחן (test/validation).
2. K fold Cross Validation.
3. הפחתת תכונות (features reduction). שימוש בתכונות עם קורלציה גבוהה ביניהן עלולה לגרום התאמת יתר.

נבצע Cross Validation בכל אחד מהמודלים, תוך חלוקה של הנתונים ל-5 קבוצות. ה-Cross Validation מתבצע כאשר כל פעם קבוצה אחרת מחמשת הקבוצות משמשת כקבוצת המבחן ויתר 4 הקבוצות משמשות לאימון הנתונים. ככל ש-Cross validation גדל, המשמעות היא שיש פחות התאמת יתר לנתוני האימון. דיאגרמה של Cross Validation [[11]](#footnote-11).



איור 7. Cross-Validation

כמו-כן, נשאף לצמצם את וקטור התכונות (features), כך שנבחר את אותן התכונות המנבאות טוב יותר את הדיאגנוזה.

# הכנת הנתונים וכלי לסטטיסטיקה וניתוח (נמצאים גם בקבצי הקוד)

## הכנת הנתונים לבדיקתם – במודלי למידת מכונה (גרסה ראשונה):**[[12]](#footnote-12)**

ייבוא ספריית Sklearn (ייבוא הספריות בנספח 3.5)

בחירת וקטור תכונות:

# select features

features=list(dataset.columns[0:31])

#features list to be used

features.append(dataset.columns[-1])

#add target values

dataset = dataset[features]

נוכל לשנות את וקטור התכונות ולבחור את התכונות בהן נשתמש בכל אחד המודלים. למשל, אם נבחר את הטורים 0:10, נגביל את וקטור התכונות כך שיכלול את תכונות הממוצע בלבד (רדיוס ממוצע, היקף ממוצע וכיו"ב.

נורמליזציה של הנתונים וחלוקת לנתוני אימון (train) ונתוני מבחן ((test (הקוד מופיע בנספח 3.6):

בכל מודלי למידת המכונה, חילקנו את הנתונים לאחר ביצוע shuffle (ערבוב רנדומלי של השורות) ל-70% נתוני אימון ו-30% נתוני מבחן וביצענו נרמול על ערכי התכונות. את הסיווג "B" שיננו ל-0 ואת הסיווג של "M" שינינו ל-1 (Encoding categorical data values).

## הכנת פונקציה המחשבת ומשרטטת "מטריצת הטעויות" ((confusion\_matrix – במודלי למידת מכונה (גרסה ראשונה) (הקוד בנספח 3.7)

הפונקציה משרטטת את ה-confusion matrix של נתוני המבחן (test), בהתבסס על תחזית המודל הנבחר על פי נתוני האימון (train). הפונקציה מדפיסה את הערכים הבאים:

* **Accuracy** - היחס בין כלל הניבויים הנכונים על נתוני המבחן לבין סה"כ נתוני המבחן.
* **Misclassification -** היחס בין כלל הניבויים השגויים על נתוני המבחן לבין סה"כ נתוני המבחן.
* **Sensitivity -** היחס בין הניבויים הנכונים ל"ממאיר" לבין כלל הנשים בעלות גידול ממאיר בפועל- הרגישות לגילוי גידול ממאיר. ככל והיחס שואף ל-1, המשמעות היא שרק מעט נשים חולות "מפוספסות" ומנובאות כבריאות.
* **Specificity** - היחס בין הניבויים הנכונים ל"שפיר" לבין כלל הנשים הבריאות. ככל והיחס שואף ל-1, המשמעות היא שרק מעט נשים בריאות מנובאות בטעות על ידי המודל כנשים חולות.
* [**False omission rate**](https://en.wikipedia.org/wiki/False_omission_rate)**-** שיעור הנשים שאובחנו באופן שגוי כחולות (על אף היותן בריאות) מכלל הנשים שאובחנו כחולות.

ככל שנקבל ערכי **Accuracy**, **Sensitivity** ו-**Specificity** גבוהים השואפים ל-1 , וערכי **Misclassification** ו-[**False omission rate**](https://en.wikipedia.org/wiki/False_omission_rate) שואפים ל-0, נוכל לומר שהמודל "טוב יותר".

## הכנת הנתונים לבדיקתם – במודלי למידה עמוקה (גרסה שנייה- מעודכן ומותאם) (הקוד בנספח 3.8)

נורמליזציה של הנתונים וחלוקת לנתוני אימון (train) ונתוני מבחן ((test: (הקוד בנספח 3.9)

בשני מודלי הלמידה העמוקה, חילקנו את הנתונים לאחר ביצוע shuffle (ערבוב רנדומלי של השורות) ל-87% נתוני אימון ו- 13% נתוני מבחן וביצענו נרמול על ערכי התכונות. את הסיווג "B" שינינו ל-0 ואת הסיווג של "M" שינינו ל-1 (Encoding categorical data values).

## הכנת פונקציה המחשבת ומשרטטת "מטריצת הטעויות" ((confusion\_matrix – במודלי למידה עמוקה (גרסה שנייה- מעודכן ומותאם): (הקוד בנספח 3.10)

הפונקציה העדכנית מציגה לנוחיותנו המון מדדים שונים ומגוונים נוספים לבדיקת איכות המודל ובדיקת הסטטיסטיקה בהתבסס על פרדיקציה (תחזית) על סטים של נתונים כמו סט הוולידציה/בדיקה.

חוץ מלצייר את המטריצה ולתת אפשרות להדפיס את ערכי המדדים להרצה כלשהי, הפונקציה אף מחזירה אובייקט שלdata frame ומחוץ לפונקציה ניתן לעשות בעזרתו חיבור של פונקצית plot לכדי סט מידע, שאיתו ניתן להציג גרפים וכו'... דבר זה הינו בשביל ההרצות ומקרי הבדיקה.

נסביר שני מדדים עיקריים אותם הצגנו ב- data frame בנוסף ל- Accuracy ולשאר המדדים שכבר הסברנו למעלה:

1. בפונקציה יש אפשרות להוסיף plot נוסף של גרף המציג את עקומה אופיינית למסווג, או עקומת ROC שהינה גרף המציג את הביצועים של מסווג דו-ערכי, לאור סף ההחלטה שנקבע לו. בגרף ניתן לראות את הקשר True Positive Rate (sensitivity) כתלות ב- False Positive Rate ( .(Fall-Out

ניתוח ROC מספק כלים לבחירת מבחן (או מסווג) אופטימלי. מרחב ROC מוגדר על ידי שני מספרים אלו- FPR בציר האופקי, ו- TPR בציר האנכי, ומבטא את העלות (במונחי שח"כ) והתועלת (במונחי שח"א) למבחן מסוים. במרחב זה, כל מבחן מיוצג כנקודה. האלכסון מחלק את מרחב ROC לשני חלקים: נקודות מעליו מייצגות סיווג מוצלח (עדיף על בחירה אקראית), ונקודות מתחתיו סיווג רע (גרוע מבחירה אקראית). ROC AUC הינו "השטח מתחת לעקומה", שמהווה המדד לטיב המודל.

2. F1 score- ממוצע הרמוני של ה- precision וה- sensitivity.

# למידת מכונה

## **[[13]](#footnote-13)**מודל Logistic Regression

התאוריה של המודל:

רגרסיה לוגיסטית משמשת לחיזוי של משתנה קטגורי- (0- שפיר, 1-ממאיר).

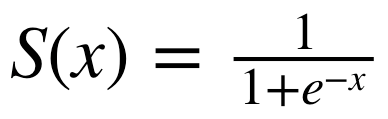
הפונקציה הלוגיסטית נתונה בנוסחה:

פונקציית הסיגמואיד s(x)

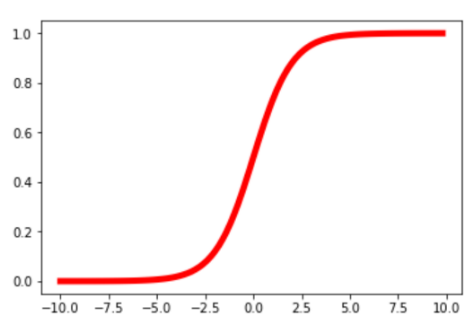
xהוא ערך האקטיבציה.

theta = np.asarray([,theta1, theta2,…..,thetan])

x = theta.dot(X\_test.T) + theta0



לפונקציית הסיגמואיד צורה של עקומת S.



פונקציית הסיגמואיד שואפת ל- 0 כאשר x שואף למינוס אינסוף ושואפת ל1 כאשר x שואף לפלוס אינסוף. *ערך פונ' הסיגמואיד בנקודה x=0 הוא 0.5. לכן, כאשר ערך הפונקציה גדול מ0.5 נסווג את הגידול כממאיר וכאשר ערך הפונ' קטן מ0.5 נסווג את הגידול כשפיר. לדוגמא, אם ערך פונ' הסיגמואיד הוא 0.65 המשמעות היא סיכוי של 65% שהגידול ממאיר. על כן, פונקציית הסיגמואיד משמשת לא רק לסיווג עצמו אלא גם על מנת לחשב את ההסתברות לקבל כל אחת מ- 2 הקלסיפיקציות.* [[14]](#footnote-14)

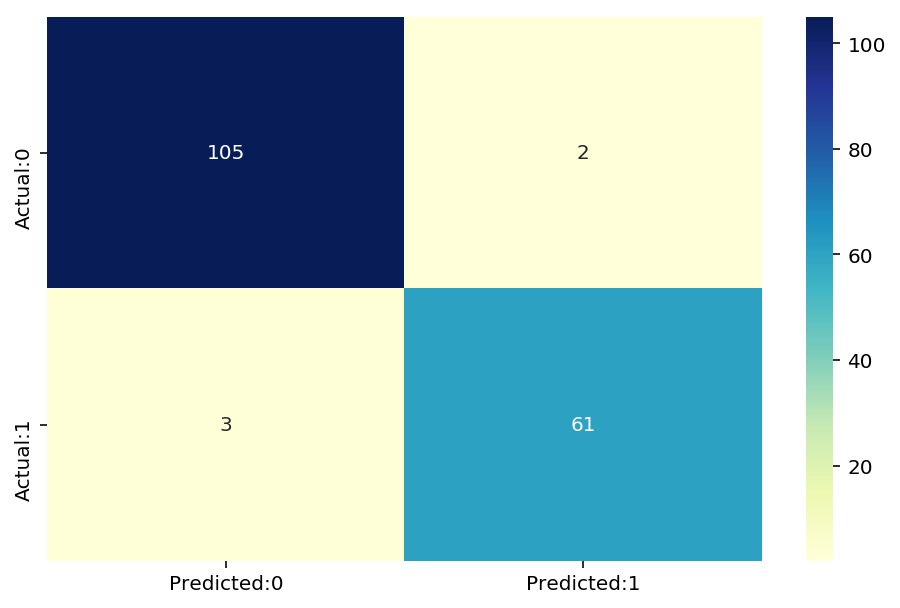
בדיקת המודל על נתוני המבחן באמצעות מודל Logistic Regression:

פונקציית לאימון הנתונים בעזרת מודל Logistic Regression (הקוד בנספח 3.11)

הרצת המודל על כלל התכונות (30 תכונות):

Accuracy= 0.971 Sensitivity= 0.953 Specificity= 0.981 False omission rate= 0.02778

Confusion Matrix:



Cross validation (הקוד נספח 3.12):

נחלק את הנתונים ל-5 קבוצות לצורך ביצוע k-fold-cross-validation:

cross\_valid\_score(), average= 0.950 std= 0.014

אם נחלק את הנתונים ל-5 קבוצות, אזי הדיוק של ה- Cross validation יהיה בממוצע כ-95%, עם סטיית תקן של כ-1.4%. הרצנו את המודל 1000 פעמים, להלן ריכוז התוצאות:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [False omission rate](https://en.wikipedia.org/wiki/False_omission_rate) | Specificity | Sensitivity | Accuracy |  |
| 0.027 | 0.980 | 0.954 | 0.970 | Average |
| 0.014 | 0.012 | 0.023 | 0.011 | Std |

מסקנות:

על אף ריבוי התכונות, המודל מנבא בדיוק גבוה מאוד בכל ההרצות. הדיוק של המודל: כ-97%, סטיית התקן: כ-1% בלבד. המודל מאתר בדיוק של מעל ל-95% את הנשים החולות, סטיית התקן: כ-2.3%. המודל מאתר בדיוק של קרוב ל-98% את הנשים הבריאות, סטיית התקן: כ-1.2%. ייתכן שהמודל מאתר טוב יותר את הנשים הבריאות, בגלל חוסר האיזון בכמות הנתונים של "בריאות" ו"חולות" בבסיס הנתונים.

מציאת התכונות המנבאות טוב יותר ודירוג התכונות לפי יכולת הניבוי שלהן:

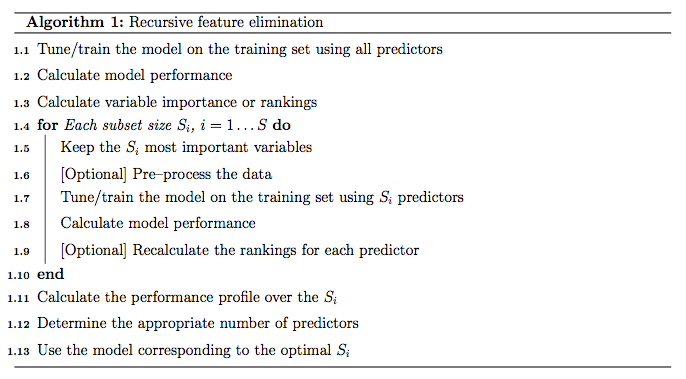
נצמצם את מספר התכונות:

פונקציית דירוג התכונות (יש לבחור את מספר התכונות) (הקוד נספח 3.13):

על מנת לצמצם את התכונות, עשינו שימוש בספרייה, from sklearn.feature\_selection import RFE.

בשיטת RFE (Recursive Feature Elimination) נבחרות התכונות המנבאות ביותר (הטובות ביותר) באופן רקורסיבי, תוך שימוש במספר תכונות הולך ופוחת בכל אינטרציה (כל פעם מנפים את התכונות המנבאת פחות טוב מכולם).

אלוגריתם ה- RFE פועל כדלקמן[[15]](#footnote-15):



צמצום התכונות ל-10 תכונות:

10 התכונות עם הדירוג הגבוה המתקבלות ב-10 הרצות הן (מתקבלות תוצאות זהות בכל 10 ההרצות):

concave points.mean radius.std area.std compactness.std radius.w texture.w perimeter.w area.w concavity.w concave points.w

הרצת המודל 1,000 פעמים על 10 התכונות הנ"ל (הקוד בנספח 3.14)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [False omission rate](https://en.wikipedia.org/wiki/False_omission_rate) | Specificity | Sensitivity | Accuracy |  |
| 0.030 | 0.986 | 0.947 | 0.972 | Average |
| 0.015 | 0.011 | 0.025 | 0.010 | Std. |

מסקנה: מתקבלות תוצאות דומות עם 10 התכונות המנבאות הכי טוב לתוצאות שמתקבלות עם כל 30 התכונות.

הדיוק של המודל דומה – כ97% (סטיית תקן של כ-1%).

המודל מאתר בדיוק של קרוב ל-95% את הנשים החולות (מעט נמוך יותר מאשר בניבוי לפי 30 תכונות), סטיית תקן: 2.6%.

המודל מאתר בדיוק של מעל ל-98% את הנשים הבריאות (מעט גבוה יותר מאשר בניבוי לפי 30 תכונות), סטיית תקן: 1.1%.

צמצום ל-5 תכונות בעלות הדירוג הגבוה ביותר:

להלן התכונות:

concave points.mean radius.w texture.w area.w concave points.w

הרצנו את המודל 1,000 פעמים על 5 תכונות:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [False omission rate](https://en.wikipedia.org/wiki/False_omission_rate) | Specificity | Sensitivity | Accuracy |  |
| 0.026 | 0.980 | 0.955 | 0.971 | Average |
| 0.014 | 0.012 | 0.024 | 0.010 | Std |

מסקנה:

מתקבלות תוצאות דומות גם עם 5 התכונות הנ"ל, כך שניתן להסתפק ב-5 תכונות אלה על מנת לנבא האם הגידול יהיה ממאיר או שפיר.

הדיוק של המודל דומה – כ97% (סטיית תקן של כ-1%).

המודל מאתר בדיוק של קרוב ל-96% את הנשים החולות (מעט גבוה מהבדיקות לגבי 30 ו-10 תכונות), סטיית תקן: 2.4%.

המודל מאתר בדיוק של מעל ל-98% את הנשים הבריאות, סטיית תקן: 1.2%.

צמצום ל-3 תכונות בעלות הדירוג הגבוה ביותר:

להלן התכונות:

concave points.mean perimeter.w area.w

הרצנו את המודל 1,000 פעמים על 3 תכונות:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [False omission rate](https://en.wikipedia.org/wiki/False_omission_rate) | Specificity | Sensitivity | Accuracy |  |
| 0.062 | 0.963 | 0.890 | 0.936 | Average |
| 0.021 | 0.017 | 0.036 | 0.015 | Std |

הדיוק של המודל יורד בצורה ניכרת בכל הפרמטרים ובפרט ב-Sensitivity וב- False omission rate. המסקנה היא שהתכונות שהורדנו במעבר בין 5 תכונות ל-3 תכונות נותנות מידע חשוב לניבוי, שאי אפשר לוותר עליו.

שינוי ערך ה- threshold :

כדי לאתר יותר נשים חולות, ניתן ל"שחק" עם ה- threshold ולשנות אותו כך שכל אישה עם הסתברות גבוהה מ-threshold נמוך יותר מ-0.5, בהתאם לקביעתנו, תסווג כ"חולה".

דוגמא ל-1000 הרצות עם שינוי ה- threshold ל-0.3 תוך שימוש ב-10 התכונות הטובות ביותר- (הקוד בנספח 3.15)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [False omission rate](https://en.wikipedia.org/wiki/False_omission_rate) | Specificity | Sensitivity | Accuracy |  |
| 0.017 | 0.958 | 0.971 | 0.963 | Average |
| 0.011 | 0.019 | 0.019 | 0.012 | Std |

שינוי ה- thresholdל-0.3 מביא לכך שהמודל מאתר בדיוק של 97% את מספר הנשים החולות. לעומת זאת יש ירידה בשיעור הדיוק של הנשים הבריאות (כ-96%).

## [[16]](#footnote-16):K-Nearest Neighbors model (KNN)

מודל זה מניח שפרטים מאותו הסוג נמצאים בסביבה סמוכה.

האלגוריתם של המודל**[[17]](#footnote-17)**:

שלב 1:

חשב את מרחקו של פרט הנבדק מכל אחד מהפרטים האחרים בבסיס הנתונים.

שלב 2:

מצא את K השכנים הקרובים לו.

שלב 3:

סווג את הפרט לפי הסוג של רוב השכנים הקרובים מתוךK השכנים הקרובים.



היתרון של המודל הוא בהיותו פשוט ליישום. החיסרון שלו שהוא איטי ככל וכמות הנתונים גדלה. עבור k=1 המודל בד"כ הינו בלתי יציב. אין מספר אופטימלי של שכנים שמתאים לכל סוגי בסיסי הנתונים. לכל סט נתונים יש מספר אופטימלי של שכנים שמתאים לו. כאשר מבצעים את הסיווג לפי מספר מועט של שכנים ה"רעש" יהיה גדול יותר ועלול להשפיע על הדיוק. לעומת זאת, אם משתמשים במספר יותר מידי גדול של שכנים, החישוב איטי ואורך זמן.

בעזרת הפונקציה הבאה חישבנו מהו מספר השכנים, בין 3-15, שנותן את הדיוק הגדול ביותר ביותר (הקוד בנספח 3.16)

Cross Validation (הקוד בנספח 3.17):

אם נחלק את הנתונים ל-5 קבוצות, אזי הדיוק של ה- Cross validation יהיה בממוצע כ-93%, עם סטיית תקן של כ-1.85%, כלומר, דיוק נמוך יותר מאשר במודל ה-Logistic Regression.

הרצת המודל 1000 פעמים- (בכל הרצה נבחר להפעיל את המודל עם מספר השכנים הקרובים שייתן דיוק מקסימלי):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [False omission rate](https://en.wikipedia.org/wiki/False_omission_rate) | Specificity | Sensitivity | Accuracy |  | No. of features |
| 0.038 | 0.993 | 0.933 | 0.971 | Average | 30 |
| 0.028 | 0.992 | 0.950 | 0.976 | Average | 10 |
| 0.039 | 0.991 | 0.932 | 0.969 | Average | 5 |
| 0.017 | 0.007 | 0.028 | 0.011 | Std | 30 |
| 0.015 | 0.007 | 0.025 | 0.010 | Std | 10 |
| 0.016 | 0.008 | 0.028 | 0.011 | Std | 5 |

גם מודל זה מנבא בדיוק רב.

במודל זה עדיף להשתמש ב-10 תכונות.

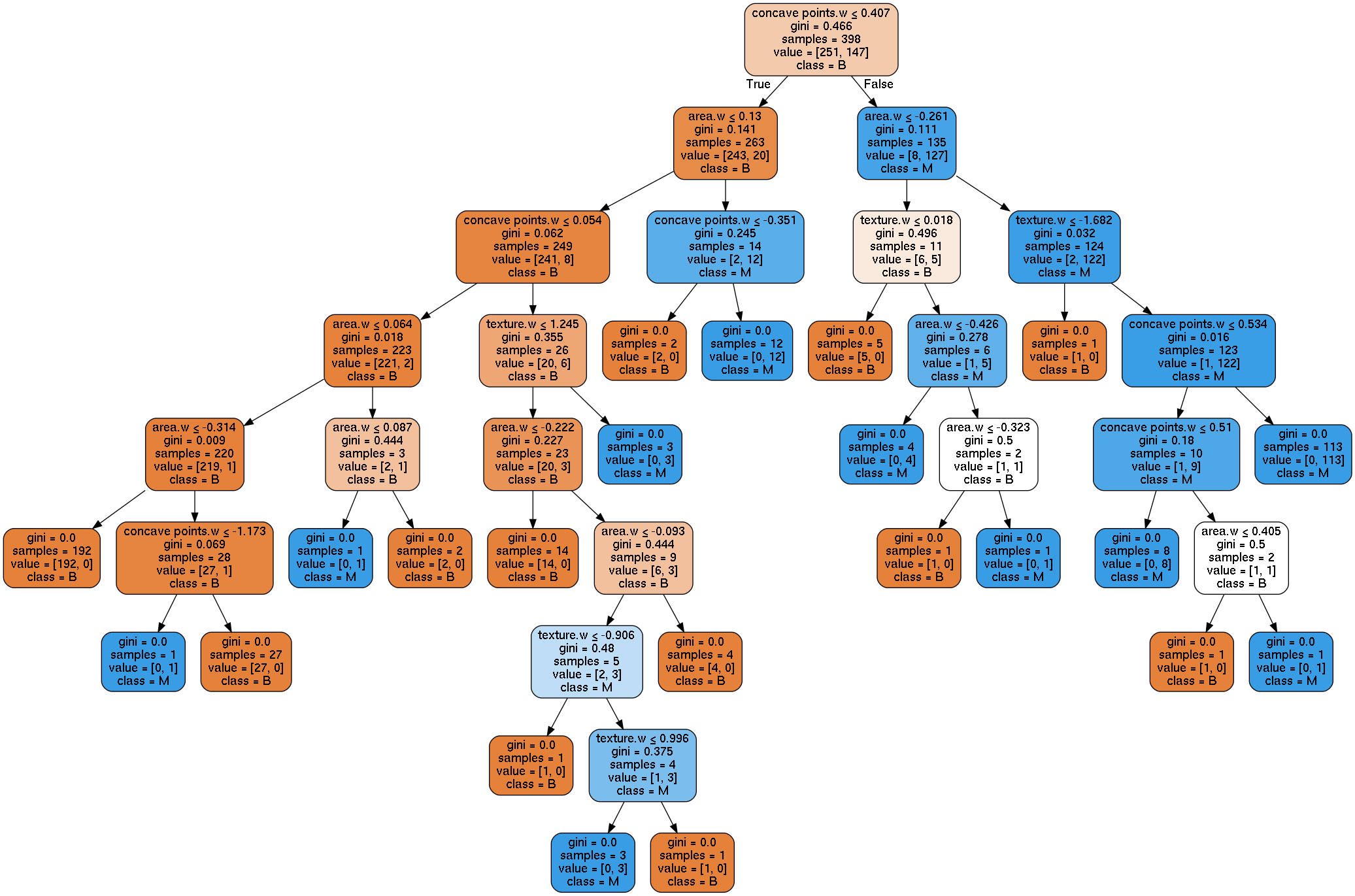
שימוש ב-5 תכונות בלבד גורם לירידה בדיוק ולאיתור מופחת של נשים שיש להן גידול ממאיר.

שימוש במודל זה עם 10 התכונות המנבאות ביותר, מביא לתוצאות דומות למודל ה- logistic regression

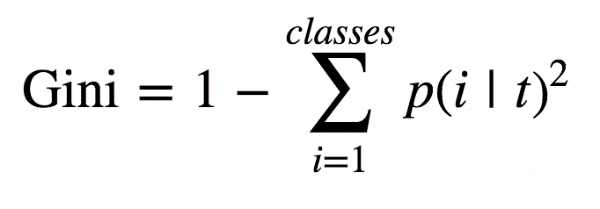
שני המודלים מביאים בממוצע לאיתור של כ-95% מהנשים נושאות הגידול הממאיר. בשני המודלים כמעט כל הנשים הבריאות מנובאות כבריאות, במודל ה-KNN מאותרות 99% נשים הבריאות כבריאות.

## [[18]](#footnote-18):Decision Tree Classifier model (DT)

האלגוריתם של המודל, על נתוני האימון, עם נתוני האימון ל המודל, עם 5 התכונות הטובות:



בכל צומת, נבחרת התכונה הטובה ביותר לחלוקת הצומת. מהעץ הנ"ל אנו רואים שהתכונה הטובה יותר לחלוקה היא concave points.w כי חלוקת הצומת הראשונה היא על בסיס תכונה זו.

[[19]](#footnote-19)

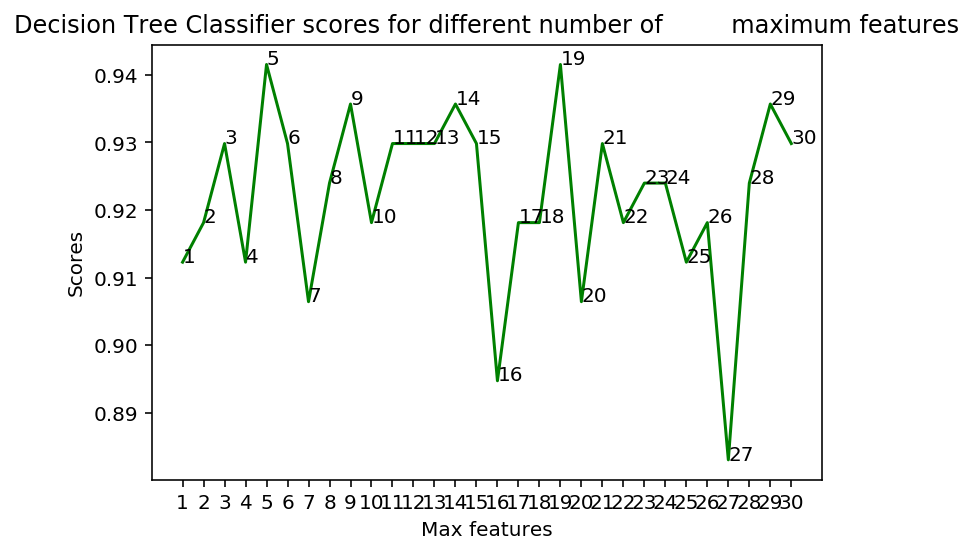
ה- Giniהוא מדד פיזור סטטסיטי שערכו נע בין 0 ל-1 המודד את מידת "אי הטוהר"impurity של כל צוצת-node. צומת שתפוצל לקטגוריות שונות היא לא טהורה- impure ואילו צומת שבסופה רק קטגוריה אחת הינה טהורה-pure.

על מנת לחשב את ערכו של Gini, נבדוק את הסיכוי להמצאות כל אחת מהקטגוריות אחרי כל node. נסכם את הריבועים של הסיכויים הנ"ל ונפחית את הסכום המתקבל מ-1. כאשר יש סיכוי אחרי הצומת (node) לקבל קטגוריה אחת בלבד, ערך ה-Gini יהיה 0, כיוון שהסיכוי למצוא את הקטגוריה הזו הוא 1. במקרה כזה הגענו ל"עלה" – leaf, הואיל ואין צורך לבצע פיצולים נוספים.

ניתן לראות בתרשים מעלה שרוב ההפרדה בין שתי הקבוצות נעשית כבר בצומת הראשונה, קטגוריזציה מלאה מתחילה כבר בצומת השלישית (Gini=0)- בה מופרדות רוב הדגימות גידול ממאיר (M), הנתונים האחרונים מקוטלגים אחרי 10 צמתים.

הפונקציה המשרטטת את ה- Decision Tree (הקוד בנספח 3.18).

חישוב דיוק המודל עבור כפונקציה של מספר התכונות (הקוד בנספח 3.19).



מהגרף ניתן להסיק את המסקנות הבאות:

בהרצה הספציפית הדיוק המקסימלי הוא כ-94%, כאשר מספר התכונות המקסימליות הינו או 5 או 9.

מספר התכונות המקסימליות משפיע מאוגד על הדיוק. לפימים הוספת תכונה נוספת, מקטינה מאוד את הדיוק. התוצאות משתנות מאוד בכל הרצה ואינן יציבות.

פונקצית המודל (הקוד בנספח 3.20).

Cross Validation

עבור 5 קבוצות, מתקבל ערך של cross validation ממוצע של כ-95%.

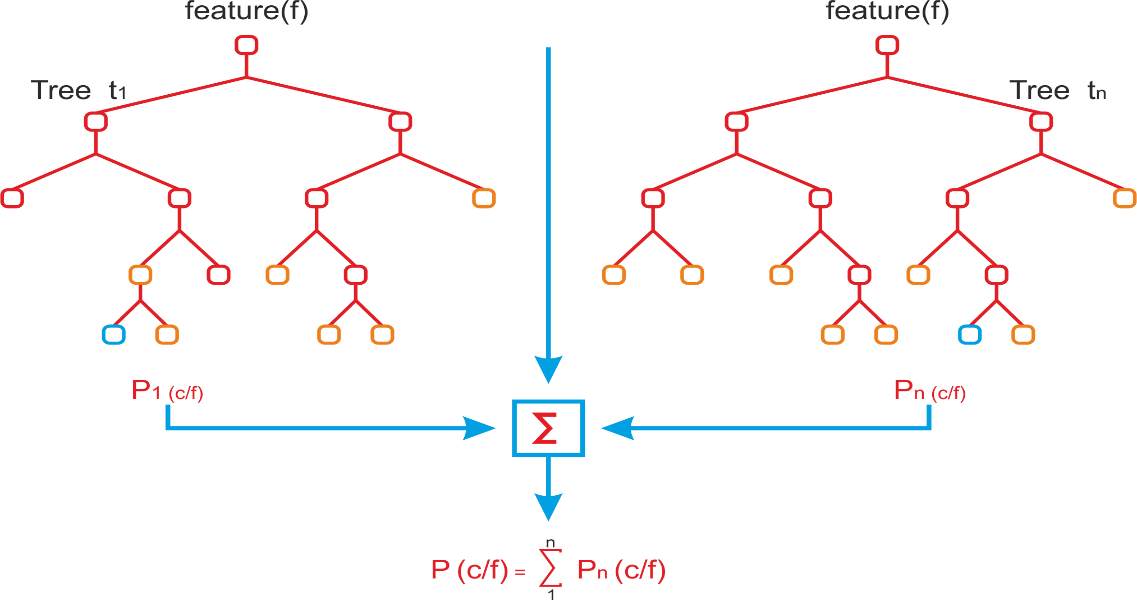
הרצנו את המודל 1,000 פעמים על 10 התכונות הטובות ביותר:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [False omission rate](https://en.wikipedia.org/wiki/False_omission_rate) | Specificity | Sensitivity | Accuracy |  |
| 0.054 | 0.944 | 0.907 | 0.931 | Average |
| 0.023 | 0.025 | 0.039 | 0.020 | Std |

מתקבלות רמות דיוק נמוכות יותר בכל הפרמטרים לעומת המודלים הקודמים.

## [[20]](#footnote-20):Random Forest Classifier model (RF)

האלגוריתם של המודל[[21]](#footnote-21):



במודל זה, במקום לחפש את התכונה הטובה ביותר לחלוקת כל צומת, נבחרת באופן רנדומלי תכונה אחת מבין מספר התכונות שנגדיר לחלוקת כל אחד מהעצים. רנדומליות החלוקה יוצרת מודל מאוזן יותר עם פחות רעש של התאמת יתר – over-fitting. ככל שנגדיל את מספר העצים נצמצם את התאמת היתר. היתרון העיקרי של המודל הוא צמצום התאמת יתר.

מספר העצים נשלט על ידי ההיפר פרמטר n\_estimators.

מספר התכונות המקסימלי בו יתחשב המודל בחלוקת כל צומת נקבע על ידי ההיפר פרמטר max\_features.

הקלסיפיקציה של נתוני המבחן נעשית לפי "מקסימום ההצבעות" של כל העצים. לדוגמא, אם בחרנו 10 עצים ביער, ש-6 מהם הצביעו על גידול "שפיר" ו-4 מהם הצביעו על גידול "ממאיר", האשה תסווג כנושאת גידול ממתאיר בהסתברות של 0.6.

החיסרון של המודל הוא שכדי לקבל תוצאות מדויקות יותר, יש צורך להגדיל את ערכי ההיפר פרמטרים הנ"ל, דבר שגורר חישוב איטי.

פונקצית המודל (הקוד בנספח 3.21).

Cross Validation

עבור 5 קבוצות, מתקבל ערך של cross validation ממוצע של כ-96%.

הרצנו את המודל 1,000 פעמים עם כל התכונות, 100 עצים ומקסימום תכונות לחלוקת כל צומת 10:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [False omission rate](https://en.wikipedia.org/wiki/False_omission_rate) | Specificity | Sensitivity | Accuracy |  |
| 0.038 | 0.972 | 0.935 | 0.958 | Average |
| 0.0195 | 0.0168 | 0.0327 | 0.0150 | Std |

רמת הדיוק בכל הפרמטרים גבוהה בהרבה לעומת מודל ה-decision tree.

מטרת עבודה זו הינה לנבא ברמת דיוק גבוהה, בהינתן סט של תכונות ((features שהתקבלו מתוך נתוני סריקה דיגיטלית של רקמת שד של נשים, האם האישה הנבדקת בריאה או נושאת גידול ממאיר, וזאת באמצעות מודלים שונים של למידת מכונה.

בעוד ה-linear regression מאתר בממוצע כ-96% מהנשים החולות, מודל ה-random forest מאתר רק כ-94% מהנשים החולות, עם סטיית תקן גבוהה יחסית של 3%.

מציאת התכונות החשובות ביותר של המודל:

פונקציה שמוצאות את התכונות החשובות ביותר במודל (הקוד בנספח 3.22):

הרצת הפונקציה:

find\_features\_importance(10)

array([23, 20, 22, 27, 7, 2, 0, 13, 3, 24])

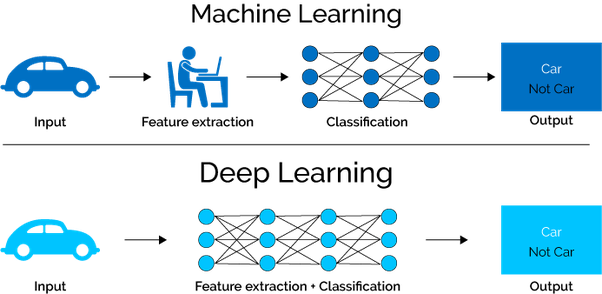
# למידה עמוקה

## רקע תיאורטי

למידה עמוקה הוא תת תחום בלמידת מכונה שהושפע רעיונית ממבנה ואופן הפעולה של המוח והנוירונים ממנו הוא מורכב.

כמו בלמידת מכונה, המטרה של למידה עמוקה היא ללמוד מתוך דוגמאות באופן עצמאי.

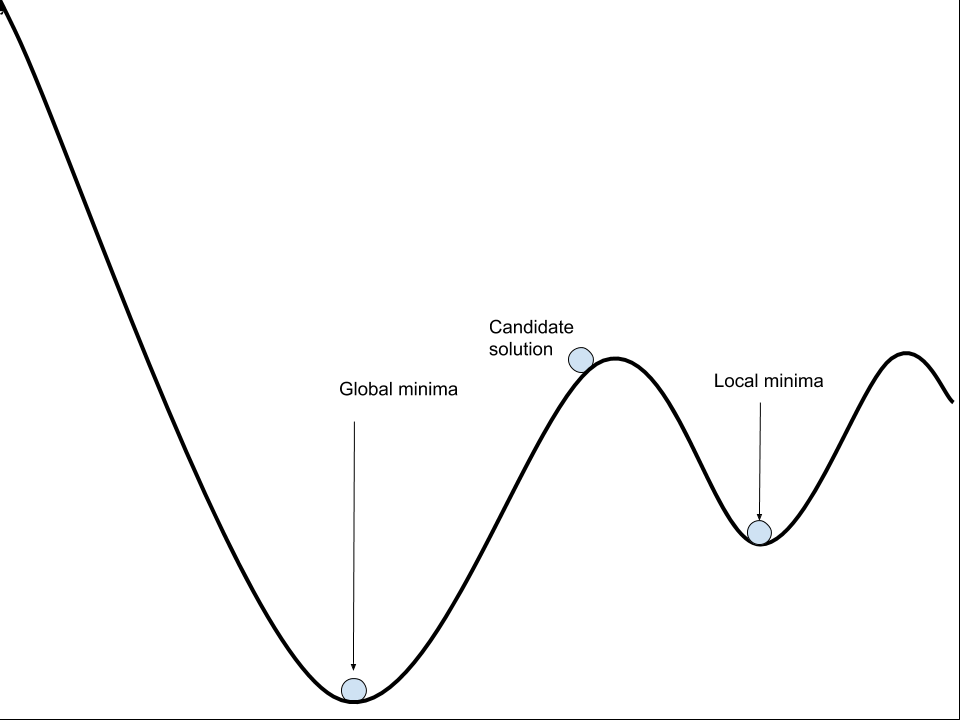
בשונה מלמידת מכונה, למידה עמוקה מתבצעת מקצה לקצה (End to End), ללא צורך בהנדסת מאפיינים של הבעיה. למעשה, רשת הנוירונים מוצאת ייצוגים חכמים עבור הקלט ובאמצעות ייצוגים אלו היא מבצעת את מטרתה.

[[22]](#footnote-22)[[23]](#footnote-23)

בנוסף בגלל שרשת הנוירונים מכילה אקטיבציות לא לינאריות היא מהווה פונקציה לא לינארית וניתן בעזרת פונקציה שכזו להביע מרחב גדול מאד של פונקציות רציפות.

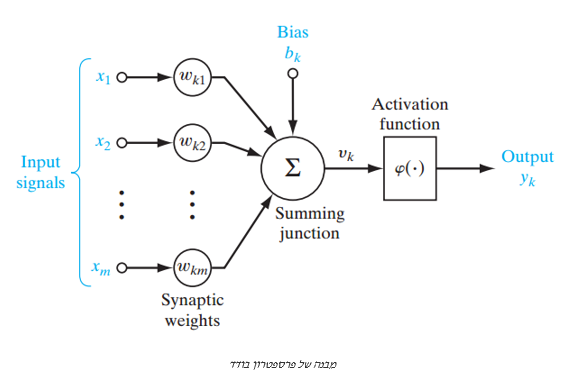
כתוצאה מכך, לרשת יש חופש לבחור מתוך מרחב היפותזות/פתרונות גדול בהרבה מאשר כל אחד מאלגוריתמי למידת המכונה. בלמידה עמוקה לא מכריכים את הפתרון להיות מסוג מסוים (למשל רגרסיה לינארית), אלא נותנים לרשת למצוא פתרון טוב לבעיה.

בלמידת עמוקה הבעיה אותה אנו מנסים לפתור אינה בעיה קמורה (convex) ואין אלגוריתם ידוע על מנת שנגיע לפתרון האופטימלי. למרות זאת, אנו מנסים לבצע מינימיזציה של פונקציית המחיר (Loss) שאותה אנו מגדירים עבור הבעיה כאילו מדובר באופטימיזציה של בעיה קמורה **[[24]](#footnote-24)**.



כדי להימנע ממצבים של התאמת יתר (overfitting) אנו מסתכלים תמיד על סט דוגמאות שעליו איננו מתאמנים (validation set) ובאמצעות תוצאות האימון ביחס לסט זה אנו מחליטים מתי להפסיק את האימון. לאחר האימון נוכל להעריך את טיב הרשת באמצעות בדיקה על סט דוגמאות נפרד (test set) **[[25]](#footnote-25)**. אימון הרשת מתבצע באמצעות שימוש באלגוריתם של ירידה במורד פונקציית המחיר (gradient descent)[[26]](#footnote-26) על מנת להגיע לנקודת מינימום (לוקאלית- מקומית). כדי לתקן גם את השכבות הנסתרות ברשת בין שכבות הקלט והפלט, אנו משתמשים בתהליך של פעפוע לאחר (backpropagation).

רשת הנוירונים מתבססת על יחידת בסיס הנקראת פרספטרון מלאכותי אשר מקבל קלט, ממשקל אותו על ידי מכפלה בווקטור משקלים (weights), מוסיף אליו ערך סף/היסט (bias) ומבצע על הערך המתקבל פונקציית אקטיבציה. הפרספטרון יחזיר כפלט את ערך האקטיבציה.

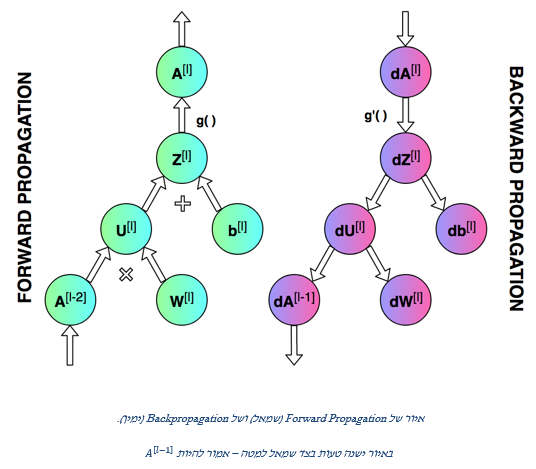


## Forward-Propagation

הרשת מבצעת forward propagation – היא מקבלת קלט ומגלגלת אותו דרך השכבות השונות אחת אחרי השניה עד שמתקבל הפלט בשכבה האחרונה.

## אלגוריתם Backpropagation

אלגוריתם זה מתבסס על חישוב הנגזרת של פונקציית המחיר ביחס לפרמטרים הפנימיים (weights, biases) של הפרספטרון אותו אנו רוצים לעדכן. אם היינו מבצעים חישוב זה באופן פשוט הרי שבשביל לעדכן כל שכבה היינו צריכים לחשב מחדש את הגרדיאנטים בין פונקציית המחיר לשכבה. דבר זה היה יוצר הרבה מאד חישובים שנעשים פעמים רבות ולכן פעולה באופן זה הייתה לוקחת זמן רב מאד. אלגוריתם backpropagation נועד כדי למנוע חישובים מיותרים במהלך חישוב הגרדיאנטים וזאת בעלות נוספת של שמירת ערכי ביניים בכל שכבה.

[[27]](#footnote-27)

אלגוריתםbackpropagation מתבסס על חישוב הגרדינאטים עבור השכבה הנתונה באמצעות כלל שרשרת. עבור כל שכבה נסמן: הקלט A ערך לפני אקטיבציהZ ופונקציית אקטיבציה g (ראה תמונה לעיל). הפרמטרים של השכבה הם המשקולות W וערכי ההיסט b. עבור כל שכבה מתקיים

פונקציית המחיר היא תמיד פונקציה של ערך אמת והפלט מהרשת (נסמן שכבה זו באינדקס

, [[28]](#footnote-28)). נסמן:

כעת נחשב את השינוי בפונקציית המחיר L ביחס לפרמטרים של שכבת הפלט:

האיבר הוא ידוע כי פונקציית המחיר ידועה מראש וגם האיבר ידוע כי הוא תלוי רק בפונקציית האקטיבציה של השכבה.

באופן דומה עבור ערכי ה- bias נקבל:

עכשיו נסתכל על השכבה השניה ():

ושוב עבור ה- bias:

קיבלנו שעבור שכבה אנחנו צריכים רק את הערך משכבה ואת הקלט () שקיבלנו משכבה . מה שיותר חשוב זה שאין שום קשר לשכבות אחרות ואת זה אלגוריתם backpropagation מנצל.

בכל שכבה k נשמור את הקלט לשכבה () ואת הערך של השכבה לפני אקטיבציה (). את הערכים האלו אנחנו שומרים בזמן המעבר קדימה (Forward-propagation). אחר כך נחשב את , כאשר הערך הראשון הוא ואז נתחיל להתגלגל אחורה כאשר כל שכבה מוסרת לזו שלפני את הערך של שחושב בתוכה.

מכיוון שאנחנו רוצים לחשב את מידת ההתאמה בין הסתברויות שהרשת נותנת כחיזוי לבין ערך האמת נשתמש בפונקציית Kullback-Leibler Divergence שמודדת "מרחק" בין התפלגויות. החלק של פונקציית זו שתלוי בחיזוי של הרשת נקרא Cross-Entropy. מכיוון שמדובר במצב בינארי נוכל לכתוב זאת באופן הבא (Binary Cross-Entropy):

כאשר הסכימה מבוצעת על כל הדוגמאות. פונקציית המחיר מגיעה למינימום 0 כאשר לכל הדוגמאות מתקיים . תהליך האופטימיזציה ישאף להביא את החיזוי קרוב ככל האפשר לערכי האמת על מנת שהמחיר יהיה קטן בשאיפה להגיע ל- 0.

הנגזרת של פונקציית המחיר היא:

נשתמש בערך זה כדי להתחיל את תהליך ה- backpropagation כאשר .

אי הליניאריות של הרשת נובעת מפונקציות האקטיבציה. אנו השתמשנו בשתי פונקציות כדי לחשב את ערכי האקטיבציה:

מכיוון שהרשת חוזה סיכוי להיות או לא להיות בסיכון לסרטן – נשתמש בפונקציית Sigmoid בשכבת הפלט כדי לקבל מספר בתחום [0..1] אותו נוכל לפרש כהסתברות להיות משויך לקבוצת הסיכון. החיזוי הבינארי הסופי יתבצע על ידי השוואת ההסתברות שהרשת חוזה לערך סף של 0.5 כאשר הסתברות של מעל לערך זה תצביע על שיוך לקבוצת הסיכון.

## תהליך האופטימיזציה – Gradient Descent

כדי להביא את פונקציית המחיר למינימום נשתמש באלגוריתם backpropagation כדי לקבל את גודל התיקון () בכל שכבה. לאחר מכן נעדכן את הפרמטרים (W, b) באמצעות הכלל הבא

כאשר הוא קצב הלימוד. ניתן להשתמש בכללי עדכון מתוחכמים יותר אשר משנים את קצב הלימוד ומתחשבים בשינויים הקודמים (דבר זה נקרא מומנטום) והם אמורים לבצע התכנסות חלקה יותר לפתרון.

## מבנה התוכנה והאלגוריתם החישובי (Layer , Network)

חילקנו את המידע כך שסט הוולידציה מהווה כבערך 13% מהמידע.

כפי שראינו בחלק התיאורטי המבנה הבסיסי ממנו מורכבת הרשת הוא שכבה בודדת. שכבה זו צריכה לשמור את הפרמטרים שלה (W, b) וכן את הקלט והפלט (A, Z). בנוסף השכבה מכילה פונקציית אקטיבציה וכן את הנגזרת של פונקציית האקטיבציה. בקוד אנחנו הטמענו 2 פונקציות אבל יש אפשרות והסבר כיצד להטמיע פונקציות נוספות. הקוד עצמו מכיל הסברים מפורטים על כל שורת קוד ודוגמאות להרצה. מאד קל לתפעל את הקוד כרשת גנרית.

כדי לממש את השכבה הגדרנו מחלקה Layer. מחלקה זו מכילה את הפרמטרים, וערכי הקלט והפלט. בזמן אתחול מופע של המחלקה ניתן לבחור באחת מפונקציות האקטיבציה שממומשות וכן את מספר הנוירונים של השכבה. בנוסף, ניתן לאתחל את המשקלים למצב נתון אחרת הם מאותחלים באופן רנדומלי.

המחלקה Layer מכילה 3 פונקציות אשר משמשות לביצוע forward propagation, backpropagation ועדכון (update) של הפרמטרים (W, b) על פי כלל העדכון שהוצג בחלק התיאורטי.

def \_\_init\_\_(self, input\_shape, units=1, bias=None, weights=None, activation=’relu’)

def forward(self, input)

def backward(self, dA)

def update(self, learning\_rate)

רשת נוירונים היא אוסף של שכבות אשר מעבירות מידע ביניהן. כדי לממש זאת הגדרנו מחלקה Network שאחראית על ניהול התקשורת בין השכבות השונות וכן על ביצוע תהליך הלמידה (train) והחיזוי (predict). כדי לבצע זאת המחלקה מקבלת מידע ומעבירה אותו משכבה לשכבה כך שכל שכבה יודעת רק על עצמה וכל התקשורת מבוצעת באמצעות מחלקת Network. כמו כן פונקציית המחיר (loss) והנגזרת שלה מוגדרות רק עבור Network.

למחלקה Network ישנה פונקציה (add\_layer) אשר מאפשרת להוסיף שכבות במהלך בניית הרשת ומבצע בדיקת תאימות בין גדלי השכבות.

תהליך הלמידה מבוצע על ידי קריאה ל-train והוא ממומש על ידי ביצוע צעדי אימון (train\_step) שכל אחד מהם מורכב משלושה צעדים:

1. מעבר קדימה דרך כל שכבות הרשת (forward).
2. חישוב פונקציית המחיר (loss) כדי להציג אותו למשתמש וחישוב של הנגזרת (loss\_derivative) כדי לבצע תיקונים בפרמטרים של השכבות.
3. מעבר אחורה דרך כל שכבות הרשת (backward) תוך כדי פניה לשכבות שיבצעו תיקון לעצמן, בהתבסס על הערך הנגזרת של פונקציית המחיר.

את תהליך הלמידה ניתן לבצע עם גודל batch לפי רצון המשתמש וכן ניתן לעצור את האימון (early\_stopping) כאשר מגיעים לרמת דיוק מספקת. לאחר מעבר על כל דוגמאות האימון (epoch) מספר פעמים שרוצים (epochs) ניתן לבצע ערבוב (shuffle) של הדוגמאות. בנוסף, ניתן לבצע לאחר כל epoch בדיקה של איכות האימון על סט ולידציה ((validation\_data שמייצג מערך עם ערכי הדוגמאות של התכונות וערכי המטרה האמתיים) באמצעות ביצוע פרדיקציה והשוואה בין הפרדיקציה לנתוני האמת. מטריצת הטעות בגרסה העדכנית שלה תבצע הערכה של האיכות והדיוק לפי מדדים שונים.

class Network:

def \_\_init\_\_(self, seed=None):

def add\_layer(self, layer):

def forward(self, input):

def loss(self, y\_true, y\_pred):

def loss\_derivative(self, y\_true, y\_pred):

def backward(self, y\_true, y\_pred, learning\_rate):

def train\_step(self, input, y\_true, learning\_rate):

def train(self, training\_feature\_samples, training\_target\_samples, epochs, learning\_rate=0.1,

batch\_size=1, validation\_data=None, shuffle=True, early\_acc\_stop=0.99, verbose=True)

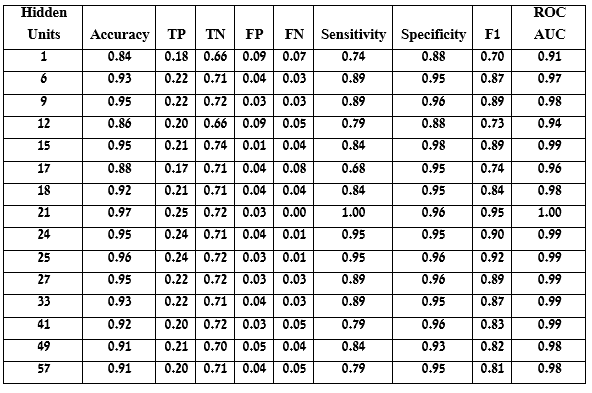
def predict(self, X)

def display\_weights\_bias(self)

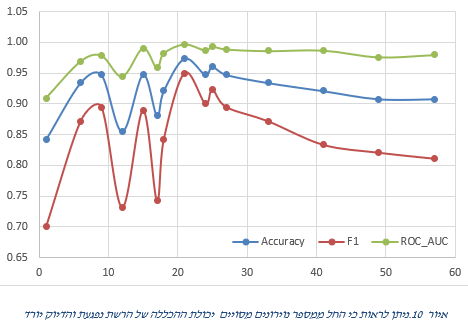
## תוצאות

הבעיה אותה אנחנו מעוניינים למדל היא בעיית סיווג בינארית ולכן שכבת הפלט של רשת הנוירונים תהיה תמיד בעלת נוירון אחד ואקטיבציית Sigmoid על מנת שנוכל לפרש הערך המתקבל כהסתברות של הקלט להיות שייך לקבוצת הסיכון. כל התוצאות מוצגות עבור סט הוולידציה.

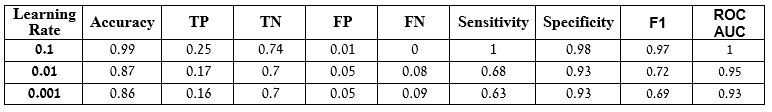
### טיב הרשת כתלות במספר הנוירונים עבור שכבה נסתרת אחת

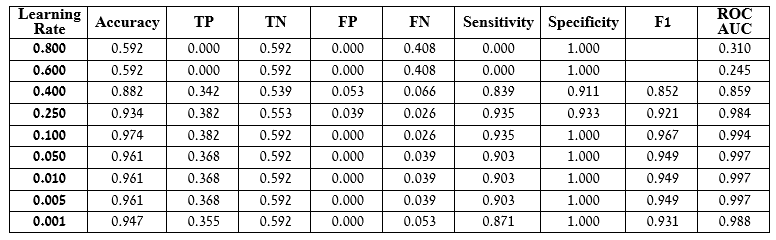


על מנת לבדוק את איכות הלמידה כתלות במספר הנוירונים, בנינו רשת בעלת שכבה נסתרת אחת עם אקטיבציית ReLU, קצב למידה 0.01 וגודל batch שהינו 1, ושינינו את מספר הנוירונים בשכבה זו. ניתן לראות שכאשר מספר הנוירונים קטן מדי (למשל – נוירון אחד) אין אפשרות לרשת ללמוד היטב. הרשת מגיעה לשיא הלמידה בערך ב-21 נוירונים וכאשר מספר הנוירונים גדול מדי הלמידה נפגעת שוב.

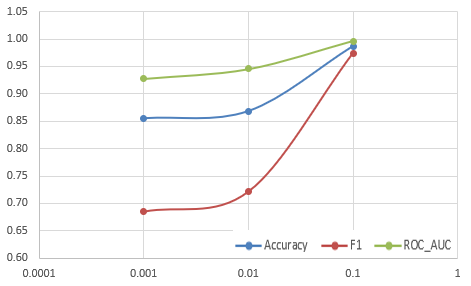


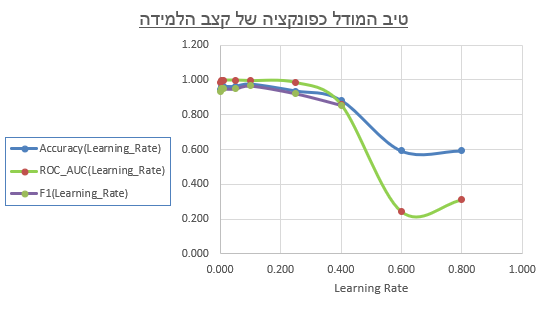
### טיב הרשת כתלות בקצב הלמידה שתי שכבות נסתרות





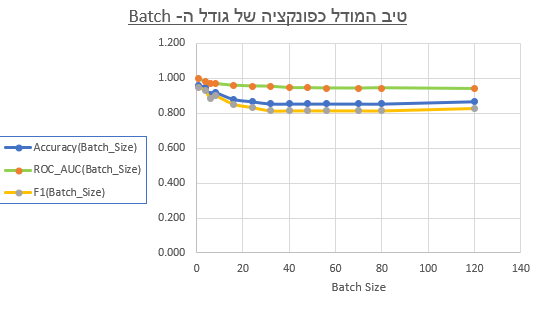
בחנו רשת בעלת שתי שכבות נסתרות: שכבה ראשונה בעלת 30 נוירונים ואקטיבציה ReLU ושכבה שנייה בעלת 30 נוירונים עם אקטיבציית Sigmoid. גודל הbatch- הוא 1 והרשת רצה מקסימום של 120 מעברים על המידע (epochs), לאחר מכן ביצענו הרצה על 1200. ניתן לראות כי ערכים קטנים מדי של קצב הלמידה אינם מאפשרים התכנסות של הרשת לנקודה בה הדיוק יהיה גבוה. דבר דומה קורה גם כאשר קצב הלמידה גבוה מדי והרשת לא מצליחה להתכנס לפתרון. ניתן לראות שבהתחלה כשקצב הלמידה נמוך ומגדילים אותו טיפה, הדיוק מטפס עד לנקודה בה הוא בערך קבוע. לאחר מכן כשמגדילים את קצב הלמידה עוד ובהגעה לקצב למידה גדול מ- 0.2, הדיוק צונח עד לנקודה בה המודל לא לומד בכלל/לא לומד נכון. לאחר שהוא צונח הדיוק נשאר נמוך סטטיסטית אך יש קורה שהקלטים גדולים מידי/לא תואמים המודל ואז יש חילוק באפס או ביצוע פעולה דומה שאינה מוגדרת. כמו כן, פעולות שאינן מוגדרות יכולות לקרות כאשר קצב הלמידה גבוה עד גבוה מאד באופן יחסי.



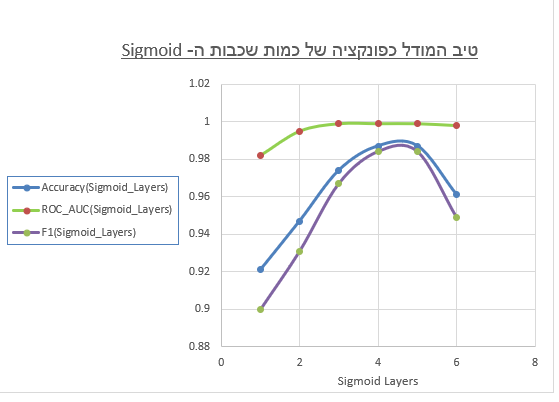


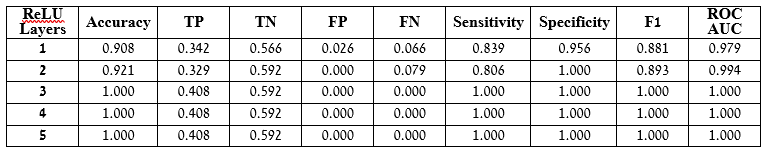
### טיב הרשת כתלות בגודל ה-Batch על הסט עבור שתי שכבות נסתרות

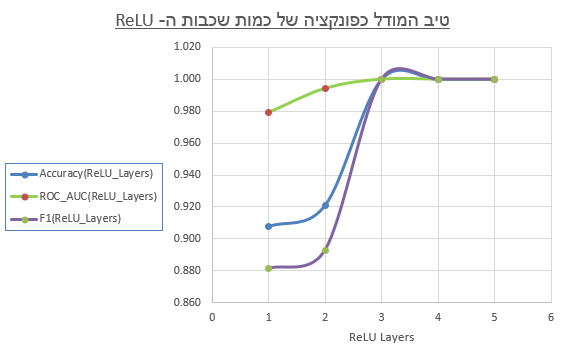
בדקנו את הרשת מהסעיף הקודם עם קצב למידה של 0.01 וגודל batch משתנה. הרשת רצה מקסימום של 1200 epochs עם קריטריון עצירה מוקדמת כאשר הדיוק מגיע לרמה של מעל ל- 99%. ניתן לראות כי גודל batch קטן נותן תוצאות טובות יותר בלימוד על בסיס הנתונים. הדיוק דועך כשמגדילים את גודל ב- batch עד שהוא מגיע בקירוב לגודל אחיד.

בנוסף, בהשוואה לסעיף הקודם (השפעת קצב הלמידה – 0.01) ניתן לראות כי לאחר הרבה צעדים גם באמצעות קצב למידה של 0.01 הרשת מצליחה ללמוד אלא שזה לוקח יותר זמן. לאחר 120 צעדים הרשת הגיעה לדיוק של כ-87% אולם לאחר 1200 צעדים היא מגיעה לדיוק של 96% (ומעלה- לפי הרצות נוספות). מכיוון שהצעדים הם קטנים יותר אפשר להגיע באופן מדויק יותר לנקודת המינימום אולם זה לוקח יותר זמן- יותר מחזורי למידה.

### טיב הרשת כתלות בכמות השכבות (המשתמשות בפונקציות אקטיבציה זהות)

בדקנו את הרשת עם מספר משתנה של שכבות (כולל שכבת הפלט) עם פונקציית Sigmoid, קצב למידה של 0.01 וגודל batch 1. הרשת רצה מקסימום של 1200 epochs עם קריטריון עצירה מוקדמת כאשר הדיוק מגיע לרמה של מעל ל- 99%. ניתן לראות כי כאשר מגדילים את כמות השכבות גדל בהדרגה הדיוק על לנקודה בה הדיוק מתחיל לקטון. ניתן לראות שבמקרה זה מספר שכבות לא גדול במיוחד, נותן דיוק טוב יותר בלימוד על בסיס הנתונים. מספר נמוך יותר של שכבות גורם לכך שזמן העיבוד יהיה קצר יותר וכך בעצם צריך יכולת עיבוד נמוכה יותר, עם פחות פרספטרונים בסך הכל.



בדקנו את הרשת עם מספר משתנה של שכבות עם פונקציית ReLU. מספר השכבות מגיע ל -5 ולא ל- 6 מכיוון שהמספר הוא סך כמות השכבות כאשר השכבה האחרונה היא תמיד Sigmoid. קבענו כמו מקודם קצב למידה של 0.01 וגודל batch 1. הרשת רצה מקסימום של 1200 epochs עם קריטריון עצירה מוקדמת כאשר הדיוק מגיע לרמה של מעל ל- 99%. ניתן לראות כי כאשר מגדילים את כמות השכבות מזנק הדיוק עד לדיוק של 100%. במקרה זה מספר שכבות ReLU לא גדול במיוחד נותן דיוק מאד טוב בלימוד על בסיס הנתונים, בזמן עיבוד קצר ובעזרת יכולת עיבוד לא גבוהה מאד.

### טיב הרשת כתלות בפונקציות האקטיבציה וסדרן על הלמידה עבור שתי שכבות נסתרות

בדקנו את הרשת עם מספר קומבינציה של שכבות שמשתמשות בפונקציות אקטיבציה בסדר שונה. קבענו קצב למידה של 0.01 וגודל batch 1. הרשת רצה מקסימום של 1200 epochs עם קריטריון עצירה מוקדמת כאשר הדיוק מגיע לרמה של מעל ל- 99%. ניתן לראות כי במקרה זה הסדר של השכבות ברשתות (מבחינת פונקציות אקטיבציה שונות) לא משנה את הדיוק. אך ניתן לראות שכאשר מספר השכבות נמוך כמו פה רצף של רק פונקציות Sigmoid ורצף של שילוב של פונקציות Sigmoid עם פונקציות ReLU יביאו לדיוק הטוב ביותר. בהשוואה למקודם כאשר היה מספר גדול יותר של שכבות המשתמשות בפונקציית ReLU, דבר שהגדיל את הדיוק ואף ל- 100%.

## אמינות התוצאה והשגיאה

בדקנו את תוצרי הרשת על סט הוולידציה וההנחה היא שסט זה מדמה באופן טוב את התפלגות האוכלוסייה. בהנחה שכך, התוצאות אותן אנו מציגים משקפות את יכולותיו של המודל לתת תחזיות אמינות לגבי כלל האוכלוסייה. יש לזכור כי המודל מבוסס על בסיס הנתונים שהוצג לעיל ואם בסיס נתונים זה אינו משקף היטב את התפלגות האוכלוסייה הכללית הרי שהלמידה לא תיתן חיזויים טובים לגבי דוגמאות מחוץ לבסיס הנתונים.

ניתן להעריך את טיב התוצאות על ידי בחינת שלושת הפרמטרים: Accuracy, Sensitivity, Specificity. פרמטרים אלו מעריכים את יכולותיו של המודל לחזות בדיוק את מי שחולה מתוך כלל האוכלוסייה החולה (Sensitivity) וכן לחזות את מי שבריא מתוך כלל האנשים הבריאים (Specificity). שלושת פרמטרים אלו מראים לנו למעשה את כמות השגיאות שהמודל עושה. מודלי הלמידה העמוקה שהצגנו מגיעים לרמות דיוק גבוהות מאוד עד לכדי 100% דיוק מבחינת מדדי ההערכה כגון Accuracy, Sensitivity, Specificity, F1, ROC AUC. תוצאות אלו בערך משתוות בחלק מהמקרים ואף טובות יותר מאלו שהושגו באמצעות שימוש באלגוריתמי למידת מכונה קלאסיים.

# סיכום הפרויקט ומסקנות

בעבודה זו ניסינו לנבא ברמת דיוק גבוהה, בהינתן סט של תכונות ((features שהתקבלו מתוך נתוני סריקה דיגיטלית של רקמת שד של נשים, האם האישה הנבדקת בריאה או נושאת גידול ממאיר, וזאת באמצעות מודלים שונים של למידת מכונה.

הגענו לרמות גבוהות של דיוק בכל הפרמטרים הנבדקים, Accuracy, Sensitivity – Specificity בשלושה מתוך ארבעה מודלים פשוטים בהם השתמשנו: Logistic Regression , K-Nearest Neighbors model ו- Random Forest Classifier model. בכל המודלים הנ"ל קיבלנו cross validation גבוה של 95%-96% בחלוקה ל-5 קבוצות שונות, דבר המעיד שהמודלים הנ"ל אינם לוקים בהתאמת יתר.

במודל הרביעי, מסוג Decision Tree Classifier model הגענו לתוצאות פחות טובות עקב בעיית התאמת יתר.

במודל ה- logistic regression קיבלנו את רמת דיוק הגבוהה ביותר בכל הפרמטרים, וזאת כאשר השתמשנו לאימון המודל רק ב-5 התכונות הטובות ביותר לניבוי, תוך השמטת יתר התכונות. בממוצע 98% מהניבויים על נתוני המבחן היו נכונים (עם סטיית תקן של כ-1%), 96% מהנשים הנושאות גידול ממאיר נובאו נכון ככאלה הנושאות גידול ממאיר (עם סטיית תקן של כ-2.5%), 98% מהנשים הבריאות נובאו כבריאות (עם סטיית תקן של כ-1%).

ראינו כי חמש התכונות המנבאות ביותר הן תכונות הקיצון הבאות:

concave points.mean radius.w texture.w area.w concave points.w

כבר הניתוח הראשוני, מבחינה ויזואלית של ההיסטוגרמות, חשדנו שתכונות אלה תהיינה התכונות החשובות ביותר לניבוי.

ייתכן כי המודל מנבא טוב יותר את הנשים הבריאות, בגלל שמספר הנשים הבריאות בבסיס הנתונים גדול יותר, דבר שגורר התאמת יתר (overfitting). על מנת להגדיל יותר את הדיוק, בכל הפרמטרים, יש לבנות בסיס נתונים גדול יותר ומאוזן יותר בשתי הקטגוריות.

ניסינו לשנות את קו הגבול, ה- thresholdשל פונקציית הסיגמואיד. לדוגמא, כאשר שינינו את קו הגבול ל-0.3, כך שכל אישה עם סיכוי של 0.3 לגידול ממאיר תסווג כחולה, המודל מאתר בדיוק של 97% את מספר הנשים החולות. לעומת זאת יש ירידה בשיעור הדיוק של הנשים הבריאות לכ-96%.

אפשר ללמוד מזה שיש בממוצע בקבוצות המבחן כ-3% נשים שהמודל ניבא שתהיינה חולות בהסתברות של פחות מ-0.3 ולמרות זאת נמצאו חולות. כל הזזה נוספת של ה- threshold כלפי מטה, מתחת ל-0.3, מביאה לעליה חדה במספר הנשים הבריאות שאותרו באופן שגוי כחולות.

מבחינה רפואית יש להגדיר את התועלת באיתור אישה חולה לעומת הנזק בסיווג שגוי של אישה בריאה כחולה. ההגדרה הרפואית הזו קובעת מהו קו הגבול הרצוי. נציע להגדיר את הנשים עם סיכוי לגידול ממאיר שבין 0.2 ל-0.5, ככאלה הצריכות לעבור בדיקות נוספות לשם הסיווג, או להשתמש בתכונות נוספות לשם הסיווג, כמו שינוי בערכי התוכנות מבדיקה קודמת, או שימוש בתכונות נוספות שלא דווקא נובעות מהדגימה שנלקחה מהרקמות – כגון תורשה וכיוצ"ב.

מאחר וסרטן השד נפוץ בשיעור גבוה, נציע לבנות בסיס נתונים גדול יותר מבסיס הנתונים בו השתמשנו, על מנת שנוכל לעבוד על big data אמיתי ולא על בסיס נתונים מוגבל ל-569 נבדקות בלבד.

מכל המודלים הפשוטים, במודל ה- logistic regression קיבלנו את רמת דיוק הגבוהה ביותר בכל הפרמטרים, וזאת כאשר השתמשנו לאימון המודל רק ב-5 התכונות הטובות ביותר לניבוי, תוך השמטת יתר התכונות. המודלים מנבאים טוב יותר את הנשים הבריאות ברוב המקרים, ייתכן שזה כך משום שמספר הנשים הבריאות בבסיס הנתונים גדול יותר (בסיס הנתונים לא לגמרי מאוזן), דבר שגורר להתאמת יתר (overfitting).

בחנו עבודות חישוביות שבוצעו על ידי אחרים על בסיס הנתונים על מנת להשוותם לתוצאות שלנו. חלק מהעבודות ערכו השוואה בין מודלים קלאסיים, אך הסתפקו במדד הדיוק accuracy הכללי ולא עשו תהליך של הפחתת תכונות[[29]](#footnote-29)- שם התקבלה רמת הדיוק הגבוהה ביותר דווקא במודל ה- .Random Forest Classification הניתוח שאנו ערכנו רחב יותר כי הוא עושה שימוש בכלים סטטיסטיים וגם משתמש במדדים שונים לבחינת רמת הדיוק, שיש להם חשיבות פרקטית רפואית, וזאת מעבר לרמת הדיוק הכללית. אנו סבורים שהצלחנו להוציא מהמודלים הקלאסיים שבחנו את "המקסימום האפשרי" מבחינת המדדים השונים, תוך השמטת תכונות הגוררות "התאמת יתר", כל זאת תוך האילוץ של בסיס נתונים לא גדול במיוחד ובלתי מאוזן מבחינת הקבוצות.

הראנו כי רשת נוירונים לא עמוקה במיוחד (רק 2 שכבות נסתרות) מספיקה כדי להגיע לדיוק גבוה מאוד (99%) וזאת ללא צורך בשינוי ערך הסף מולו מתבצע הסיווג. בכך מגיעה רשת הנוירונים אותה כתבנו באופן עצמאי לתוצאות טובות יותר מאשר מודלים קלאסיים של למידת מכונה.

אנו חושבים שאיסוף מאגר נתונים גדול יותר שמכיל מידע לגבי מספר רב יותר של נשים יאפשר להגיע לתוצאות יותר טובות ומדויקות מעבודתנו. הבחנו שמאגרי המידע בתחום אינם מספקים. מצאנו מאגרים המכילים יותר נשים מאשר במאגר שלנו, אך עם פחות תכונות. אנו חושבים שיצירת מאגרים שמכילים יותר נבדקות ויותר תכונות יעזרו בפיתוח תחום מחקר זה.

# נספחים:

## נספח 1- ביבליוגרפיה (לפי הפניות בעבודה)

נספח 1-ביבליוגרפיה (לפי הפניות בעבודה)

1. תמונת שער לעבודה. שנג וונג (2017). [מקור תמונת שער](https://blog.insightdatascience.com/automating-breast-cancer-detection-with-deep-learning-d8b49da17950)
2. מאגר מידע (1995). [Breast Cancer Wisconsin Diagnostic Data](https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29)

Creators:   
1. Dr. William H. Wolberg, General Surgery Dept.

University of Wisconsin, Clinical Sciences Center

Madison, WI 53792

wolberg '@' eagle.surgery.wisc.edu

2. W. Nick Street, Computer Sciences Dept.

University of Wisconsin, 1210 West Dayton St., Madison, WI 53706

street '@' cs.wisc.edu 608-262-6619

3. Olvi L. Mangasarian, Computer Sciences Dept.

University of Wisconsin, 1210 West Dayton St., Madison, WI 53706

olvi '@' cs.wisc.edu

Donor: Nick Street

1. [Machine Learning](https://he.wikipedia.org/wiki/%D7%9C%D7%9E%D7%99%D7%93%D7%AA_%D7%9E%D7%9B%D7%95%D7%A0%D7%94) (ויקיפדיה, למידת מכונה)
2. סביבת העבודה cocalc בה ביצענו את העבודה <https://cocalc.com/>
3. ראה 2
4. ראה 2
5. [Data Visualization](https://cocalc.com/projects/923776ef-943b-467b-a03b-2a8cd26507dd/files/note/Breat_cancer_Visualise-3.ipynb?session=default)  : קובץ –cocalc שבו הצגנו את הנתונים. ראה ריכוז קבצי ה- cocalc של העבודה בנספח לעבודה זו.

[Feature Selection For Machine Learning](https://machinelearningmastery.com/feature-selection-machine-learning-python/)  . ג'ייסון בראונלי(2016)

מתוך "ויקיפדיה" הגדרת המונח "התאמת יתר" [wikipedia-overfitting](https://he.wikipedia.org/wiki/%D7%94%D7%AA%D7%90%D7%9E%D7%AA_%D7%99%D7%AA%D7%A8)

<https://elitedatascience.com/overfitting-in-machine-learning> .

1. Cross validation. <https://scikit-learn.org/stable/modules/cross_validation.html>
2. קישור לcocalc. קובץ הכנת הנתונים לשימוש במודלים השונים. [Preapering\_Of\_Data](https://cocalc.com/projects/923776ef-943b-467b-a03b-2a8cd26507dd/files/note/Breat_cancer_models.ipynb?session=default)

קובץ העבודה שלנו ב-cocalc על מודל logistic regression: [Logistic Regression](https://cocalc.com/projects/923776ef-943b-467b-a03b-2a8cd26507dd/files/note/Breat_cancer_models-lr-1.ipynb?session=default)

1. **Logistic Regression Theory: An Overview-** [logistic-regression-theory](https://dzone.com/articles/logistic-regression-theory). וינאי קומאר (2017)
2. <https://topepo.github.io/caret/recursive-feature-elimination.html>  
   Recursive Feature Elimination

קובץ העבודה שלנו על מודל KNN ב-cocalc: <https://cocalc.com/projects/923776ef-943b-467b-a03b-2a8cd26507dd/files/note/Breat_cancer_models-knn.ipynb?session=default>

1. <https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn>  
   אבינש נבלני (2018)

קובץ העבודה שלנו על מודל DT ב-cocalc:

<https://cocalc.com/projects/923776ef-943b-467b-a03b-2a8cd26507dd/files/note/Breat_cancer_models-dt.ipynb?session=default>

1. <https://towardsdatascience.com/decision-tree-an-algorithm-that-works-like-the-human-brain-8bc0652f1fc6>

Decision Tree: an algorithm that works like the human brain. אמה גרימלדי (2018)

קובץ העבודה שלנו על מודל RF ב-cocalc: [RF](https://cocalc.com/projects/923776ef-943b-467b-a03b-2a8cd26507dd/files/note/Breat_cancer_models-rf.ipynb?session=default)

1. [Random Forest for churn prediction](https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781788479042/2/ch02lvl1sec23/random-forest-for-churn-prediction)

יער החלטה רנדומלי.

1. <https://medium.com/@Say2neeraj/what-is-the-difference-between-machine-learning-and-deep-learning-5795e4415be9>
2. <https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063>
3. <https://machinelearningmastery.com/why-training-a-neural-network-is-hard/>
4. [אופנים להתמודדות עם overfitting במודלים של למידה עמוקה - ברט קראמנס (2018)](https://towardsdatascience.com/handling-overfitting-in-deep-learning-models-c760ee047c6e)
5. <https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e>
6. <https://towardsdatascience.com/lets-code-a-neural-network-in-plain-numpy-ae7e74410795>
7. הפנייה מספר 28 היא להערה בתחתית העמוד.
8. <https://towardsdatascience.com/building-a-simple-machine-learning-model-on-breast-cancer-data-eca4b3b99fa3>

## נספח 2 – קישורים לכל קבצי המודלים

**מודלי למידת מכונה קלאסית:**

[Data Visualization](https://cocalc.com/projects/923776ef-943b-467b-a03b-2a8cd26507dd/files/note/Breat_cancer_Visualise-3.ipynb?session=default)

[Logistic Regression](https://cocalc.com/projects/923776ef-943b-467b-a03b-2a8cd26507dd/files/note/Breat_cancer_models-lr-1.ipynb?session=default)

[KNN](https://cocalc.com/projects/923776ef-943b-467b-a03b-2a8cd26507dd/files/note/Breat_cancer_models-knn.ipynb?session=default)

[DT](https://cocalc.com/projects/923776ef-943b-467b-a03b-2a8cd26507dd/files/note/Breat_cancer_models-dt.ipynb?session=default)

[RF](https://cocalc.com/projects/923776ef-943b-467b-a03b-2a8cd26507dd/files/note/Breat_cancer_models-rf.ipynb?session=default)

**מודלי למידה עמוקה:**

[מודל Keras](https://cocalc.com/share/ec0cc296-50a7-40c8-97ea-283524b1f48a/Project/Project%20updated.ipynb?viewer=share) גנרי

[מודל רשת נוירונים גנרי](https://share.cocalc.com/share/ec0cc296-50a7-40c8-97ea-283524b1f48a/Project/Project%20updated%20part%202.ipynb?viewer=share)

## נספח 3- הפניות לקוד בתוך המסמך (ההפניות מסומנות בעבודה בירוק)

**נספח 3.1 – ייבוא ספריות פייתון לצורך הצגת הנתונים**

#import libraries

import numpy as np # To do calcutaions with ndarrays

import pandas as pd # To create a confusion matrix data frame

import seaborn as sn # To create a heatmap from the confusion matrix

import matplotlib.pyplot as plt # To visualize the model and the confusion matrix

from matplotlib import rcParams

from matplotlib.cm import rainbow

%matplotlib inline

import statsmodels.api as sm

import scipy.stats as st

import warnings

warnings.filterwarnings('ignore')

**נספח 3.2 – הצגת הנתונים**

#split dataframe into two based on diagnosis

dfM=dataset[dataset['Diagnosis'] ==1]

dfB=dataset[dataset['Diagnosis'] ==0]

**נספח 3.3 – סרטוט ההיסטוגרמות**

plt.rcParams.update({'font.size': 8})

fig, axes = plt.subplots(nrows=5, ncols=2, figsize=(8,10))

axes = axes.ravel()

for idx,ax in enumerate(axes):

ax.figure

binwidth= (max(dataset[features[idx]]) - min(dataset[features[idx]]))/50

bins=np.arange(min(dataset[features[idx]]), max(dataset[features[idx]]) + binwidth, binwidth)

ax.hist([dfM[features[idx]],dfB[features[idx]]], bins,alpha=0.5,stacked=True, normed = True, label=['M','B'], color = ['b', 'r'])

ax.legend(loc='upper right')

ax.set\_title(features[idx])

plt.tight\_layout()

plt.show()

**נספח 3.4 – מטריצת קורלציה**

df\_corr\_worse = dataset.iloc[:, 0:10].corr()

plt.figure(figsize=(8,8))

sn.heatmap(df\_corr\_worse, cmap="Blues", annot=True)

plt.show()

**נספח 3.5 – ייבוא ספריית Sklearn**

#Sklearn library for implementing Machine Learning models and processing of data

from sklearn.model\_selection import train\_test\_split

To split the data into train data and test data#

from sklearn import preprocessing

from sklearn.preprocessing import StandardScaler

To scale (standardize) the data#

from sklearn.metrics import confusion\_matrix

To make a confusion matrix#

from sklearn.utils import shuffle

# To shuffle the data before the model starts to learn

from sklearn.preprocessing import LabelEncoder

#Encoding categorical data values

from sklearn.feature\_selection import RFE

#to select and rank features

from sklearn.model\_selection import cross\_val\_score

# to find cross validation score

from sklearn.preprocessing import binarize #in order to change threshold

ייבוא המודלים מספריית Sklrean:

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

**נספח 3.6 – נורמליזציה של הנתונים וחלוקתם לקבוצות**

#split dataframe into two based on diagnosis

dfM=dataset[dataset['Diagnosis'] ==1]

dfB=dataset[dataset['Diagnosis'] ==0]

#shuffle data to achieve randomal split between train data and test data

dataset = shuffle(dataset)

#split data to features values and Diagnosis value

X = dataset.iloc[:, :-1].values

Y = dataset.iloc[:, -1].values

#Encoding categorical data values

labelencoder\_Y = LabelEncoder()

Y = labelencoder\_Y.fit\_transform(Y)

#split our dataset into training and testing datasets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size = 0.3, random\_state = 0)

#Standardize features by removing the mean and scaling to unit variance

scaler = StandardScaler().fit(X\_train)

X\_train = scaler.transform(X\_train)

X\_test = scaler.transform(X\_test)

**נספח 3.7 – הכנת פונקציה המחשבת ומשרטטת "מטריצת הטעויות" ((confusion\_matrix – במודלי למידת מכונה (גרסה ראשונה)**

# this function draws the confusion matrix and prints Accuracy, Sensitivity, Specificity and False omission rate

def draw\_confusion\_matrix(y\_test,y\_pred):

cm=confusion\_matrix(y\_test,y\_pred)

conf\_matrix=pd.DataFrame(data=cm,columns=['Predicted:0','Predicted:1'],index=['Actual:0','Actual:1'])

TN=cm[0,0]

TP=cm[1,1]

FN=cm[1,0]

FP=cm[0,1]

Accuracy=round((TN+TP)/float(TN+TP+FN+FP),3)

Misclassification=1-Accuracy

Sensitivity=round(TP/(float(TP+FN)),3)

Specificity=round(TN/(float(TN+cm[0,1])),3)

False\_omission\_rate=round(FN/(FN+TN),5)

print ('Accuracy=',Accuracy,'Sensitivity=', Sensitivity, 'Specificity=', Specificity,' False Omission Rate=' , False\_omission\_rate)

plt.figure(figsize = (8,5))

sn.heatmap(conf\_matrix, annot=True,fmt='d',cmap="YlGnBu")

**נספח 3.8 – הכנת הנתונים לבדיקתם – במודלי למידה עמוקה (גרסה שנייה- מעודכן ומותאם)**

# Importing packages

import numpy as np # To do calculations with ndarrays

from random import shuffle as rndShuffle # To shuffle the indices of the train samples and train targets while training the model

import pandas as pd # To create a confusion matrix data frame

import matplotlib.pyplot as plt # To visualize the model and the confusion matrix

import seaborn as sn # To create a heatmap from the confusion matrix

# Importing Sklearn modules for processing data

from sklearn.utils import shuffle # To shuffle the data before the model starts to learn

from sklearn.model\_selection import train\_test\_split # To split the data into train data and test/validation data

from sklearn.preprocessing import StandardScaler # To scale/normalize (standardize) the data (by fitting to train samples)

from sklearn.metrics import confusion\_matrix, roc\_curve, auc # To make a confusion matrix

# To set the backend of matplotlib to the 'inline' backend: With this backend, the output of plotting commands is displayed inline within frontends

%matplotlib inline

dataset = pd.read\_csv('/home/user/data/Breast\_Cancer\_Data\_CSV.csv') # Reading the dataset with panda, returned as two-dimensional data structure/data frame with labeled axes

dataset.drop(['ID number'], axis=1, inplace=True) # Dropping the feature ID since it has nothing to do with predicting cancer (or so we believe)

#The data doesn't contain NaNs/nulls, therefore there's no need to drop samples

print('Cancer diagnosis dataset dimensions: {}'.format(dataset.shape)) # Printing shape size of dataset

dataset.head() # Displaying the upper part of the data frame, with the labels and numbering

dataset = shuffle(dataset) # Shuffle data to achieve randoma split between train data and test/validation data

# Split dataframe into two based on diagnosis

target\_samples\_data = dataset.pop('Diagnosis') # Popping the target values out of the dataset to targets

print('Cancer diagnosis dataset dimensions: {}'.format(dataset.shape)) # Printing shape size of dataset

feature\_samples\_data = dataset # feature\_samples becomes the new dataset which contains the feature samples without the target values

print(target\_samples\_data.unique()) # Printing the unique target elements

target\_samples\_data = target\_samples\_data.map({'M': 1, 'B': 0}) # Encoding categorical target values

print(target\_samples\_data.unique()) # Printing the new unique target elements

**נספח 3.9 – נורמליזציה של הנתונים וחלוקת לנתוני אימון (train) ונתוני מבחן ((test**

# Splitting dataset into training data and validation data consistently (due to random\_state = 0) # The validation data is the test data and vice versa

# Because we already have a big enough validation set, the test set importance decreases and we can still do a quite accurate accuracy/performance and error evaluation

training\_feature\_samples\_data, validation\_feature\_samples\_data, training\_target\_samples\_data, validation\_target\_samples\_data = train\_test\_split(feature\_samples\_data, target\_samples\_data, test\_size=0.13356, random\_state=0)

scaler = StandardScaler().fit(training\_feature\_samples\_data) # Fitting the scaler to the training data, to standard scale the rest of the data

training\_feature\_samples\_data = scaler.transform(training\_feature\_samples\_data) # Standardizing the training data

print(training\_feature\_samples\_data.mean(axis=0)) # Printing standardized data mean of the training columns

validation\_feature\_samples\_data = scaler.transform(validation\_feature\_samples\_data) # Standardizing the validation data

print(validation\_feature\_samples\_data.mean(axis=0)) # Printing standardized data mean of the validation columns

**נספח 3.10 – הכנת פונקציה המחשבת ומשרטטת "מטריצת הטעויות" ((confusion\_matrix – במודלי למידה עמוקה (גרסה שנייה- מעודכן ומותאם)**

def draw\_confusion\_matrix(y\_true, y\_pred, title='', printStats=True, plot\_figure=True):

# https://en.wikipedia.org/wiki/Confusion\_matrix

# Creating a confusion matrix, counting the true and false predictions aside the actual real predictions (the targets) - values as percentage

cm = confusion\_matrix(y\_true.ravel(), np.round(y\_pred).ravel())

# Creating a two-dimensional panda data structure/data frame with labeled axes from the confusion matrix - values are number of samples

conf\_matrix = pd.DataFrame(data=cm, columns=['Predicted:0', 'Predicted:1'], index=['Actual:0', 'Actual:1'])

# Assigning values to each prediction rate value:

total = cm.sum() # Total human subjects - women that participated

TN = cm[0, 0] / total # True Negative- Predicted Negative as a true value

TP = cm[1, 1] / total # True Positive- Predicted Positive as a true value

# https://en.wikipedia.org/wiki/Type\_I\_and\_type\_II\_errors#False\_positive\_and\_false\_negative\_rates

# https://en.wikipedia.org/wiki/False\_positives\_and\_false\_negatives#false\_negative\_rate

FN = cm[1, 0] / total # False Negative- Predicted Negative as a false value

FP = cm[0, 1] / total # False Positive- Predicted Positive as a false value, # https://en.wikipedia.org/wiki/False\_alarm

P = TP + FN # Condition Positive

N = FP + TN # Condition Negative

# https://en.wikipedia.org/wiki/Sensitivity\_and\_specificity

# https://en.wikipedia.org/wiki/Precision\_and\_recall

# https://en.wikipedia.org/wiki/Accuracy\_and\_precision

# https://en.wikipedia.org/wiki/Positive\_and\_negative\_predictive\_values

accuracy = (TN + TP)

precision = TP / float(TP + FP)

sensitivity = TP / float(P) # https://en.wikipedia.org/wiki/Hit\_rate

specificity = TN / float(N)

negative\_predictive\_value = TN / float(TN + FN)

# Misclassification = 1 - Acuuracy

# False Discovery Rate = 1 - precision, # https://en.wikipedia.org/wiki/False\_discovery\_rate

# Miss Rate = 1 - Sensitivity

# Fall Out = 1 - Specificity, # https://en.wikipedia.org/wiki/False\_positive\_rate, # https://en.wikipedia.org/wiki/Information\_retrieval#Fall-out

# False Omission Rate = 1 - Negative Predictive Value, # https://www.fxsolver.com/browse/formulas/False+Omission+Rate

prevalence = (TP + FN) # Prevalence of breast cancer among the checked group, # https://en.wikipedia.org/wiki/Prevalence

# https://en.wikipedia.org/wiki/Likelihood\_ratios\_in\_diagnostic\_testing#negative\_likelihood\_ratio

positive\_likelihood\_ratio = sensitivity / float(1 - specificity)

negative\_likelihood\_ratio = (1 - sensitivity) / float(specificity)

diagnostic\_odds\_ratio = positive\_likelihood\_ratio / float(negative\_likelihood\_ratio) # https://en.wikipedia.org/wiki/Diagnostic\_odds\_ratio

# https://en.wikipedia.org/wiki/F1\_score, # https://en.wikipedia.org/wiki/Harmonic\_mean#Harmonic\_mean\_of\_two\_numbers

F1\_score = 2 \* precision \* sensitivity / float(precision + sensitivity)

# https://en.wikipedia.org/wiki/Matthews\_correlation\_coefficient

matthews\_correlation\_coefficient = (TP \* TN - FP \* FN) / float(((TP + FP) \* P \* N \* (TN + FN))\*\*0.5)

# https://en.wikipedia.org/wiki/Youden%27s\_J\_statistic

informedness = float(sensitivity + specificity - 1)

# In statistics the concept of markedness is quantified as a measure of how much one variable is marked as a predictor or possible cause of another, and is also known as Δp (deltaP) in simple two-choice cases

markedness = float(precision + negative\_predictive\_value - 1)

# https://scikit-learn.org/stable/auto\_examples/model\_selection/plot\_roc.html

fpr, tpr, \_ = roc\_curve(y\_true.ravel(), y\_pred.ravel()) # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc\_curve.html#sklearn.metrics.roc\_curve

roc\_auc = auc(fpr, tpr) # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.auc.html#sklearn.metrics.auc

if (printStats):

print('Accuracy = %.3f Precision = %.3f Sensitivity = %.3f Specificity = %.3f\nNegative Predictive Value = %.3f Prevalence = %.3f Positive Likelihood Ratio = %.3f Negative Likelihood Ratio = %.3f Diagnostic Odds Ratio = %.3f\nF1 Score = %.3f Matthews Correlation Coefficient = %.3f Informedness = %.3f Markedness = %.3f ROC AUC = %.3f' % (accuracy, precision, sensitivity, specificity, negative\_predictive\_value, prevalence, positive\_likelihood\_ratio, negative\_likelihood\_ratio, diagnostic\_odds\_ratio, F1\_score, matthews\_correlation\_coefficient, informedness, markedness, roc\_auc)) # Printing statistic values

if (plot\_figure):

fig, axes = plt.subplots(1, 2, figsize=(20, 8))

sn.heatmap(conf\_matrix, annot=True, fmt='.0f', cmap='YlGnBu', ax=axes[0]) # Creating heatmap from the panda dataframe

axes[1].plot(fpr, tpr, label='ROC\_AUC (area under curve) = %.3f' % (roc\_auc)) # https://en.wikipedia.org/wiki/Receiver\_operating\_characteristic

axes[1].plot([0, 1], [0, 1], 'r--')

plt.title(title + ' | ROC AUC')

axes[1].set\_xlabel('FPR')

axes[1].set\_ylabel('TPR')

plt.legend()

plt.grid()

metrics = pd.DataFrame(data=[accuracy, TP, TN, FP, FN, sensitivity, specificity, F1\_score, roc\_auc], index=['Accuracy', 'TP', 'TN', 'FP', 'FN', 'Sensitivity', 'Specificity', 'F1', 'ROC\_AUC'])

return metrics.T

**נספח 3.11 – פונקציית לאימון הנתונים בעזרת מודל LogisticRegression (נספח 3.11)**

def logistic\_reg():

from sklearn.linear\_model import LogisticRegression

logistic = LogisticRegression()

logistic.fit(X\_train,y\_train)

# fit the model on train data

y\_pred=logistic.predict(X\_test)

#print("score=",logistic.score(X\_test, y\_test) )

#print(logistic.coef\_,logistic.intercept\_)

draw\_confusion\_matrix(y\_test,y\_pred)

return(y\_pred)

**נספח 3.12 – Cross validation לLogistic Regression**

# this function prints cross validation score of n groups

def cross\_valid\_score(n):

clf = LogisticRegression()

scores = cross\_val\_score(clf, X, Y, cv=n)

print('average=',np.average(scores),'std=', np.std(scores))

cross\_valid\_score(5)

**נספח 3.13 – פונקציית דירוג התכונות (יש לבחור את מספר התכונות) (נספח 3.13):**

def rank\_features(num):

logistic = LogisticRegression()

# create the RFE model and select num attributes

rfe = RFE(logistic, num)

rfe = rfe.fit(X\_train,y\_train)

# summarize the selection of the attributes

print(rfe.support\_)

print(rfe.ranking\_)

t=len(rfe.ranking\_)

for i in range (t-1):

if (rfe.ranking\_[i]==1):

print (features[i])

**נספח 3.14 – הרצת המודל (Logistic Regression) 1,000 פעמים על 10 התכונות הנ"ל (נספח 3.14):**

features=list(dataset.columns[[7,10,13,15,20,21,22,23,26,27]])

para=np.zeros([1000,4])

i=0

for i in range (1000):

dataset1 = shuffle(dataset)

X = dataset1.iloc[:, :-1].values

Y = dataset1.iloc[:, -1].values

labelencoder\_Y = LabelEncoder()

Y = labelencoder\_Y.fit\_transform(Y)

#Encoding categorical data values

split our dataset into training and testing datasets#

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size = 0.3, random\_state = 0)

scaler = StandardScaler().fit(X\_train)

X\_train = scaler.transform(X\_train)

X\_test = scaler.transform(X\_test)

y\_pred=logistic\_reg()

a=values\_confusion\_matrix(y\_test,y\_pred)

para[i,:]=a

i=i+1

print(np.average(para, axis=0))

print(np.std(para, axis=0))

**נספח 3.15** – שינוי ה- ה- threshold **ל-0.3 תוך שימוש ב-10 התכונות הטובות ביותר**

# change threshold

from sklearn.linear\_model import LogisticRegression

logistic = LogisticRegression()

logistic.fit(X\_train,y\_train)

# fit the model on train data

y\_pred\_prob\_yes=logistic.predict\_proba(X\_test)

y\_pred=binarize(y\_pred\_prob\_yes,n)[:,1]

**נספח 3.16 – בעזרת הפונקציה הבאה חישבנו מהו מספר השכנים, בין 3-15, שנותן את הדיוק הגדול ביותר**

def find\_max\_knn\_score():

knn\_scores = []

for k in range(3,15):

knn\_classifier = KNeighborsClassifier(n\_neighbors = k)

knn\_classifier.fit(X\_train, y\_train)

knn\_scores.append(knn\_classifier.score(X\_test, y\_test))

return (knn\_scores.index(max(knn\_scores))+3)

**נספח 3.17 – Cross Validation לKNN Model**

def cross\_valid\_score(n):

clf = KNeighborsClassifier(n\_neighbors = find\_max\_knn\_score())

scores = cross\_val\_score(clf, X, Y, cv=n)

print(scores, 'average=',np.average(scores),'std=', np.std(scores))

cross\_valid\_score(5)

average= 0.933 std= 0.019

**נספח 3.18 – הפונקציה המשרטטת את ה- Decision Tree**

# instantiating the model

tree = DecisionTreeClassifier()

# fitting the model

model = tree.fit(X\_train,y\_train)

#plot desicion tree

from sklearn.externals.six import StringIO

from sklearn.tree import export\_graphviz

import pydotplus

from IPython.display import Image

dot\_data = StringIO()

export\_graphviz(model, out\_file = dot\_data, filled=True, rounded=True, proportion=False, special\_characters=True, feature\_names=features[:-1], class\_names=['B', 'M'])

graph = pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())

Image(graph.create\_png())

**נספח 3.19 – חישוב דיוק המודל עבור כפונקציה של מספר התכונות**

def dt\_print\_max\_features():

dt\_scores = []

for i in range(1, (X.shape[1]) + 1):

dt\_classifier = DecisionTreeClassifier(max\_features = i, random\_state=0)

dt\_classifier.fit(X\_train, y\_train)

dt\_scores.append(dt\_classifier.score(X\_test, y\_test))

plt.plot([i for i in range(1, (X.shape[1]) + 1)], dt\_scores, color='green')

for i in range(1, (X.shape[1]) + 1):

plt.text(i, dt\_scores[i-1], (i))

plt.xticks([i for i in range(1, X.shape[1] + 1)])

plt.xlabel('Max features')

plt.ylabel('Scores')

plt.title('Decision Tree Classifier scores for different number of maximum features')

**נספח 3.20 – פונקציית מודל DT**

def dt\_model():

dt\_classifier=DecisionTreeClassifier(max\_features=dt\_model\_index\_max(), random\_state = 0)

dt=dt\_classifier

dt.fit(X\_train, y\_train)

y\_pred=dt.predict(X\_test)

print("score=",dt.score(X\_test, y\_test) )

draw\_confusion\_matrix(y\_test,y\_pred)

**נספח 3.21 – פונקציית מודל RF**

def rf\_model():

rf\_classifier = RandomForestClassifier(n\_estimators=100,max\_features=10)

rf=rf\_classifier

rf.fit(X\_train, y\_train)

y\_pred=rf.predict(X\_test)

print("score=",rf.score(X\_test, y\_test) )

draw\_confusion\_matrix(y\_test,y\_pred)

**נספח 3.22 – פונקציה המוצאת את התכונות החשובות ביותר במודל RF**

def find\_features\_importance(n):

clf = RandomForestClassifier(n\_estimators=100, max\_features=10)

clf = clf.fit(X\_train, y\_train)

importances = clf.feature\_importances\_

idx = (-importances).argsort()[:n]

return( idx )

BreastCancerAI (c) by Roi Dvir

BreastCancerAI is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

You should have received a copy of the license along with this work. If not, see <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

1. [מקור תמונת שער](https://blog.insightdatascience.com/automating-breast-cancer-detection-with-deep-learning-d8b49da17950) [↑](#footnote-ref-1)
2. את בסיס הנתונים ניתן להוריד מהאתר הבא: [Breast Cancer Wisconsin Diagnostic Data](https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29) [↑](#footnote-ref-2)
3. [Machine Learning](https://he.wikipedia.org/wiki/%D7%9C%D7%9E%D7%99%D7%93%D7%AA_%D7%9E%D7%9B%D7%95%D7%A0%D7%94) (ויקיפדיה, למידת מכונה) [↑](#footnote-ref-3)
4. <https://cocalc.com/> [↑](#footnote-ref-4)
5. את בסיס הנתונים ניתן להוריד מהאתר הבא: [Breast Cancer Wisconsin Diagnostic Data](https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29) [↑](#footnote-ref-5)
6. ראה הערת שוליים מספר 2 לעיל [↑](#footnote-ref-6)
7. [Data Visualization](https://cocalc.com/projects/923776ef-943b-467b-a03b-2a8cd26507dd/files/note/Breat_cancer_Visualise-3.ipynb?session=default)  : קובץ –cocalc שבו הצגנו את הנתונים. ראה ריכוז קבצי ה- cocalc של העבודה בנספח לעבודה זו. [↑](#footnote-ref-7)
8. [Feature Selection For Machine Learning](https://machinelearningmastery.com/feature-selection-machine-learning-python/)  [↑](#footnote-ref-8)
9. מתוך "ויקיפדיה" הגדרת המונח "התאמת יתר" [wikipedia-overfitting](https://he.wikipedia.org/wiki/%D7%94%D7%AA%D7%90%D7%9E%D7%AA_%D7%99%D7%AA%D7%A8) [↑](#footnote-ref-9)
10. <https://elitedatascience.com/overfitting-in-machine-learning> [↑](#footnote-ref-10)
11. <https://scikit-learn.org/stable/modules/cross_validation.html> [↑](#footnote-ref-11)
12. <https://cocalc.com/projects/923776ef-943b-467b-a03b-2a8cd26507dd/files/note/Breat_cancer_models.ipynb?session=default> [↑](#footnote-ref-12)
13. קובץ העבודה שלנו ב-cocalc על מודל

    [Logistic Regression](https://cocalc.com/projects/923776ef-943b-467b-a03b-2a8cd26507dd/files/note/Breat_cancer_models-lr-1.ipynb?session=default) [↑](#footnote-ref-13)
14. Logistic Regression Theory: An Overview- [logistic-regression-theory](https://dzone.com/articles/logistic-regression-theory) [↑](#footnote-ref-14)
15. <https://topepo.github.io/caret/recursive-feature-elimination.html> [↑](#footnote-ref-15)
16. קובץ העבודה שלנו על מודל KNN ב-cocalc: <https://cocalc.com/projects/923776ef-943b-467b-a03b-2a8cd26507dd/files/note/Breat_cancer_models-knn.ipynb?session=default> [↑](#footnote-ref-16)
17. <https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn> [↑](#footnote-ref-17)
18. קובץ העבודה שלנו על מודל DT ב-cocalc:

    <https://cocalc.com/projects/923776ef-943b-467b-a03b-2a8cd26507dd/files/note/Breat_cancer_models-dt.ipynb?session=default> [↑](#footnote-ref-18)
19. <https://towardsdatascience.com/decision-tree-an-algorithm-that-works-like-the-human-brain-8bc0652f1fc6> [↑](#footnote-ref-19)
20. קובץ העבודה שלנו על מודל RF ב-cocalc: <https://cocalc.com/projects/923776ef-943b-467b-a03b-2a8cd26507dd/files/note/Breat_cancer_models-rf.ipynb?session=default> [↑](#footnote-ref-20)
21. <https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781788479042/2/ch02lvl1sec23/random-forest-for-churn-prediction> [↑](#footnote-ref-21)
22. <https://medium.com/@Say2neeraj/what-is-the-difference-between-machine-learning-and-deep-learning-5795e4415be9> [↑](#footnote-ref-22)
23. <https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063> [↑](#footnote-ref-23)
24. [הקושי באימון רשת נוירונים - ג'ייסון בראונלי (2019)](https://machinelearningmastery.com/why-training-a-neural-network-is-hard/) [↑](#footnote-ref-24)
25. [אופנים להתמודדות עם overfitting במודלים של למידה עמוקה - ברט קראמנס (2018)](https://towardsdatascience.com/handling-overfitting-in-deep-learning-models-c760ee047c6e) [↑](#footnote-ref-25)
26. <https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e> [↑](#footnote-ref-26)
27. <https://towardsdatascience.com/lets-code-a-neural-network-in-plain-numpy-ae7e74410795> [↑](#footnote-ref-27)
28. ספירת השכבות היא כמו במערכים בפייתן כאשר סופרים מהסוף – האיבר האחרון הוא 1- וזה שלפניו הוא 2- וכן הלאה. [↑](#footnote-ref-28)
29. כך לדוגמא, <https://towardsdatascience.com/building-a-simple-machine-learning-model-on-breast-cancer-data-eca4b3b99fa3> [↑](#footnote-ref-29)