פרוייקט גמר בכריית מידע

מטלות מנחה 21 + 22

מגיש: קורן סנגייר

תוכן עניינים:

ממן 21 (הכנת נתונים ועצי החלטה) ...........................................2-14

ממן 22 (בייס, ניתוח אשכולות ורשתות נוירונים מלאכותיות) ..........15-20

סיכום ומסקנות .........................................................................21

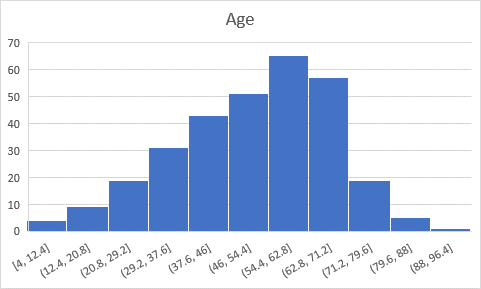
ביבליוגרפיה .............................................................................22

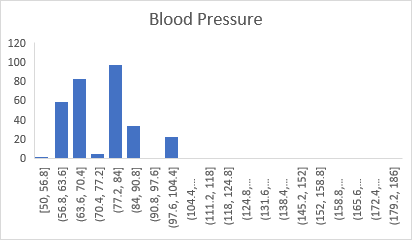
הגדרת הבעיה והכנת הנתונים

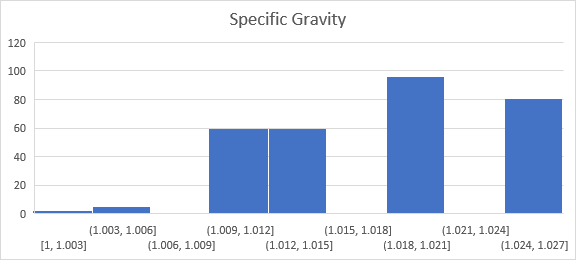
1. **מטרת כריית המידע:**מטרת כריית המידע היא לחזות האם לחולה יש CKD – Chronic Kidney Disease, על בסיס רשומות החולה ומערכת חיזוי המבוססת על 400 רשומות של פציינטים שחלקם (250) חולים והשאר (150) אינם חולים. הרשומות מכילות נתונים כגון גיל, לחץ דם, רמת המוגלובין בדם ונתונים נוספים. את כל הנתונים ופירוט עליהם ניתן לראות בסעיף ב. על מסד הנתונים שקיבלנו יש מספר הנחות יסוד.
   1. הנתונים מדויקים עד כדי תיקון ידני של תאים שזזו או ערכי שהשתנו במקצת (למשל הוספת רווח לערך yes או no).
   2. הנתונים הגיעו בפורמט arff. מכיוון שיותר נוח לעבוד עם פורמט xlsx, יש להניח שהמעבר מarff לcsv והמעבר מcsv לxlsx הוא פשוט ועמיד לטעויות.
   3. רשימת המידע על השדות מהאתר שממנו קיבלנו את מסד הנתונים מעודכנת ונכונה. (האתר מוזכר בסוף המסמך, בביבליוגרפיה).
2. **הנתונים בשימוש:**

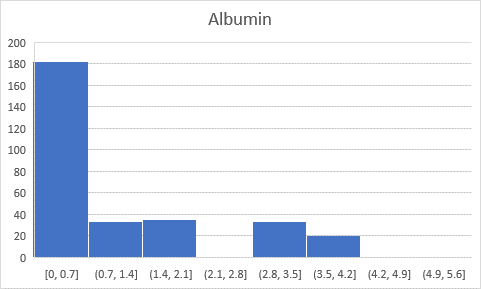
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **שם השדה** | **יחידות מידה** | **סוג הנתונים** | **תחומי ערכים** | **mean** | **median** | **mode** |
| **Age** | years | נומרי | 0-100 | 51.2 | 54 | 55 |
| **Blood pressure** | Mm/Hg | נומרי | 50-180 | 75.8 | 80 | 80 |
| **Specific gravity** |  | נומרי | 1 + 0.05k (0<=k<=5) | 1.02 | 1.02 | 1.02 |
| **Albumin** |  | קטגורי | 0-5 |  |  |  |
| **Sugar** |  | קטגורי | 0-5 |  |  |  |
| **Red blood cells** |  | בינארי | 1 אם תקין |  |  |  |
| **Pus cell** |  | בינארי | 1 אם תקין |  |  |  |
| **Puc cell clumps** |  | בינארי | 1 אם קיים |  |  |  |
| **Bacteria** |  | בינארי | 1 אם קיים |  |  |  |
| **Blood glucose random** | Mgs/dl | נומרי | 70-490 | 140.6 | 117 | 107 |
| **Blood urea** | על עמודה זו נוותר, כי רק 70 מתוך 400 הרשומות (כ17%) הוא בטווח התקין של הערכים (לפי חיפוש בגוגל), ולכן סביר שהערכים המוזנים שגויים. | | | | | |
| **Serum Creatinine** | Mgs/dl | נומרי | 0.4-10 | 1.8 | 1.2 | 1.2 |
| **Sodium** | mEq/L | נומרי | 104-163 | 138.5 | 139 | 135 |
| **Potassium** | mEq/L | נומרי | 2.5-7.6 | 4.34 | 4.4 | 5 |
| **Hemoglobin** | Gms | נומרי | 6-17.8 | 13.2 | 13.7 | 15 |
| **Packed cell volume** | Percentage | נומרי | 0-100 | 40 | 41 | 52 |
| **White blood cell count** | Cells/ cumm | נומרי | 2200-16700 | 8355 | 8100 | 9200 |
| **Red blood cell count** | Millions/ cumm | נומרי | 2.1-6.5 | 4.8 | 4.8 | 4.5 |
| **Hypertension** | על עמודה זו נוותר, כי העמודה blood pressure נותן לנו את אותו המידע באופן יותר מפורט. | | | | | |
| **Diabetes Mellitus** |  | בינארי | 1 אם חולה |  |  |  |
| **Coronary Artery Disease** |  | בינארי | 1 אם חולה |  |  |  |
| **Appetite** |  | בינארי | 1 אם יש |  |  |  |
| **Pedal Edema** |  | בינארי | 1 אם חולה |  |  |  |
| **Anemia** |  | בינארי | 1 אם חולה |  |  |  |
| **Classification** |  | בינארי | 1 אם חולה |  |  |  |

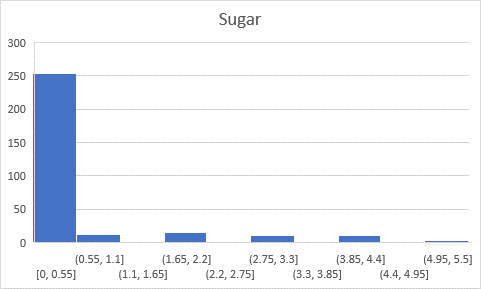
1. **שלבי הKDD:**
   1. איסוף נתונים: נעשה עבורנו (הורדנו את הנתונים מהאתר שבביבליוגרפיה).
   2. ניקוי נתונים:
      1. העברה מפורמט arff לcsv ואז לxlsx.
      2. מעבר ידני על מסד הנתונים: זיהוי בעיות הבולטות בעין (תיקון ההזזה בשורה 371).
      3. הסרת העמודות bu, hypertension.
      4. החלפת ערכים מחוץ לטווח בערך חסר.
      5. הסרת שורות מרובות חוסרים (5 או יותר ערכים חסרים – שורש מכמות השדות הקיימים).
      6. השלמת ערכים מספריים חסרים בהתאם לעמודה (ממוצע).
      7. השלמת ערכים בינאריים חסרים (עצי חיזוי פשוטים).
      8. המרת ערכים קטגוריים לבינאריים (איפה שניתן).
      9. דיסקרטיזציה וניפוי הנתונים לערכים שמישים בעזרת שיטות כמו binning – **נדחה לעת חישוב העץ.**
   3. בחירת שיטות לכריית מידע:
      1. הכנה של מסד הנתונים: excel.
      2. ניקוי הנתונים קוד פייתון בפיתוח שלי בשימוש בספריית pandas.
      3. חישוב עצי החלטה בשימוש בספריית sklearn.
      4. הצגת עצי ההחלטה בשימוש בספריית graphviz.
   4. הרצת כריית המידע
      1. שימוש בקוד פייתון מוכן מספריות שהוזכרו.
      2. עטיפה בקוד משלי להתאמות לצרכי המטלה.
   5. ניתוח התוצאות
      1. ניתוח סטטיסטי מבוסס על תוצאות הריצה.
      2. ניתוח סטטיסטי של "כלל הרוב".
      3. החלטה האם כריית המידע מוצלחת.
   6. הסקת מסקנות
2. **חלופות לכריית המידע:**
   1. מיפוי אלגוריתמים שנלמדו:
      1. עץ החלטה ID3 – Information Gain  
         העץ מורכב על ידי שימוש במדד אנטרופיה וחישוב הרווח (הצמצום) של האנטרופיה מפיצול מסויים.  
         **יתרונות:** מדד gain שימושי מאוד ונותן תוצאות טובות לכריית מידע. העץ פשוט למימוש ולבנייה.  
         **חסרונות:** information למול ratio – במקרה של עץ שאינו בינארי, תינתן העדפה לתכונות מרובות ערכים. בנוסף, העץ המתקבל עלול להיות בומבסטי ולכן איטי בהרצתו בעת החיזוי.
      2. עץ החלטה C4.5 – Gain Ratio  
         העץ פועל בדומה לID3, רק שבחישוב הרווח נכנס גם היחס לכמות הערכים השונים לכל תכונה.  
         **יתרונות:** מדד gain שימושי מאוד ונותן תוצאות טובות לכריית מידע. העץ פשוט למימוש ולבנייה, מתמודד עם בעית הinformation למול ratio על מסדי נתונים שבהם קיימות מגוון תכונות עם כמויות ערכים שונות.  
         **חסרונות:** העץ המתקבל עלול להיות בומבסטי ולכן איטי בהרצתו בעת החיזוי. לפעמים ייתן תוצאות חלשות יותר מאשר ID3 בגלל הוספת היחס בעצים בינאריים.
      3. עץ החלטה CART – Gini  
         העץ מורכב לפי מדד ג'יני, שמאפשר התמודדות טובה גם לתכונות מרובות ערכים וגם לתכונות בינאריות. לאחר הרכבת העץ, מתבצע תהליך גיזום אשר מקטין את העץ בשביל לחסוך בזמן ריצה. תהליך הגיזום רץ עד שהוא מגיע לאיזון.  
         **יתרונות:** מדד gini גם הוא חזק ונותן תוצאות טובות לכריית מידע. העץ המתקבל קטן ולכן קל יותר להריץ עליו חיזויים.  
         **חסרונות:** לוקח הרבה יותר זמן לאמן עץ כזה בגלל הגיזום - pruning. העץ קטן יותר ולכן עלול להניב תוצאות פחות טובות מID3 – מעין tradeoff עם זמן הריצה של עץ.
      4. האצה אדפטיבית AdaBoost  
         היער מבוסס על עצי CART, רק שכל אחד מהם מקבל משקל בהצבע שבסוף התהליך. בנוסף, גם בעת הלמידה ניתנים משקלים שונים לערכים שונים.  
         **יתרונות:** מדד gini גם הוא חזק ונותן תוצאות טובות לכריית מידע. היער המתקבל מורכב ממספר עצים שונים ומותאם לנתונים ברגישות גבוהה יותר.  
         **חסרונות:** לוקח הרבה יותר זמן לאמן יער כזה, גם בגלל הגיזום (במידה ובוחרים להכליל גיזום) וגם בגלל ריבוי העצים. זמן הריצה עלול להיות ארוך בעת חיזוי בגלל ריבוי העצים.
      5. יער אקראי Random Forest  
         היער מבוסס על עצי ID3, רק שכל אחד מהם מאומן על תכונות שונות. כל אחד מהם מקבל גם משקל שונה להצבעה שבחיזוי.  
         **יתרונות:** מדד gain שימושי מאוד ונותן תוצאות טובות לכריית מידע. היער המתקבל מורכב ממספר עצים שונים ומותאם לנתונים ברגישות גבוהה יותר.  
         **חסרונות:** לוקח הרבה יותר זמן לאמן יער כזה, בגלל ריבוי העצים. זמן הריצה גם הוא ארוך משמעותית בעת חיזוי בגלל ריבוי העצים הגדולים (אין גיזום).
   2. החלטה בהקשר הפרוייקט:
      1. מכיוון שרוב המסווגים הקטגוריים שלנו הם בינאריים, אין טעם ממשי להשתמש בGain Ratio על פני Information Gain, ולכן ויתרתי על C4.5.
      2. את שאר המודלים מימשתי והשוויתי בקוד, מכיוון שלכל אחד מהם יש יתרונות וחסרונות שעלולים להיות מועילים, ולכן רציתי להשוות ביניהם.
      3. בחרתי לבסוף ביער אקראי והאצה אדפטיבית, כי אפשר היה לוותר על ID3 מכיוון שהוא "מוכל" בתוך היער האקראי, ועל CART מכיוון שהוא "מוכל" בהאצה אדפטיבית. המימוש של העצים האחרים נשאר בקוד.
3. **שלבי הכנת הנתונים:**
   1. ערכים חסרים וערכים שגויים  
      במסד הנתונים התקבלו שני סוגים של נתונים בעייתיים: ערכים חסרים וערכים שגויים. התחלתי מטיפול בערכים השגויים, שאותם מצאתי בכך שהגדרתי טווחים הגיוניים לכל אחת מהשדות מחיפוש מהיר בגוגל, כל ערך שגוי הוחלף בערך חסר.  
      הטיפול בערכים החסרים התבצע בשלבים. ראשית, הסרתי את השורות שבהן חסר לפחות 5 ערכים. הגדרתי שמספר הערכים החסרים הוא לפחות שורש ממספר התכונות מכיוון שזה שלב שבו תלות גדולה מדי בין הערכים יכולה בהחלט להשפיע על החיזוי. שנית, השלמתי את הערכים המספריים החסרים בממוצע של התכונה הזו. לבסוף, את הערכים הבינאריים החסרים השלמתי בעזרת עצי חיזוי בסיסיים שהרכבתי על בסיס הערכים המספריים בעבור כל אחד מהערכים.  
      כמובן שאם היה חסר ערך בעמודת הCKD הייתי מוחק את אותה השורה כי היא לא מוסיפה לנו מידע, אבל לא היו שורות כאלו במסד הנתונים.
   2. ניקוי נתונים  
      כחלק מהאלגוריתמים השונים לבנית עצים בספריית sklearn מתבצע binning לערכים. תהליך זה ממיר מגוון גדול של ערכים לכדי טווחים, וזה מאפשר לעצים לעבוד בגרנולריות גבוהה יותר, מכיוון שלא צריך להתייחס לכל אחד מהערכים, במיוחד בשדות שבהן אין הבדל גדול בין ערכים עוקבים, כמו למשל גיל. בנוסף, הbinning מכליל את הממוצע שהכנסנו בשלב הערכים החסרים, ובכך חוסך את העיגול שהיינו צריכים לעשות לו אחרת.
   3. המרת נתונים  
      הנתונים הבינאריים היו שמורים בצורה מילולית, ובמקום זה הפכתי אותם לערכים של 0 ו1. המרה זו חשובה במיוחד במקרה של הclassification, שבו הערכים 1 ו0 והסדר שלהם קובע את ההבדל בין FP לFN. במקרה זה בחרתי ש1 יהיה שהמטופל חולה בCKD.  
      בנוסף, הסרתי את העמודות blood urea וhypertension מסיבות שמפורטות בטבלה.
   4. תצוגה גרפית  
      משום מה הexcel מתעקש לעשות binning בייצור הגרפים ולא הצלחתי לבטל את זה ועדיין שזה יראה נורמאלי

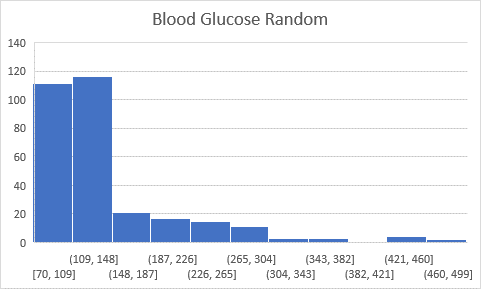


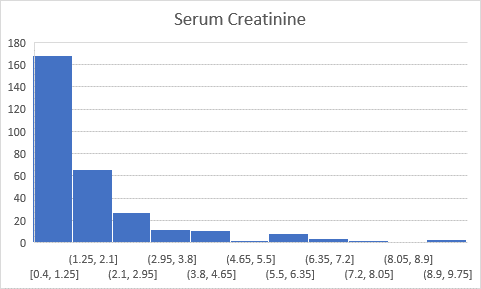


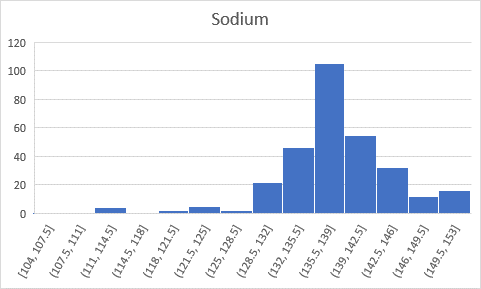


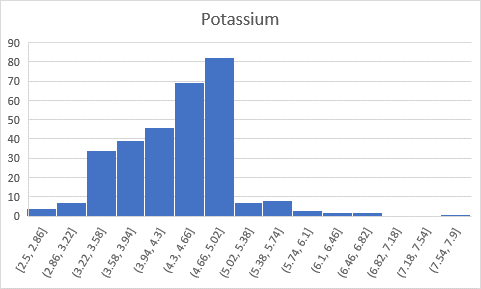


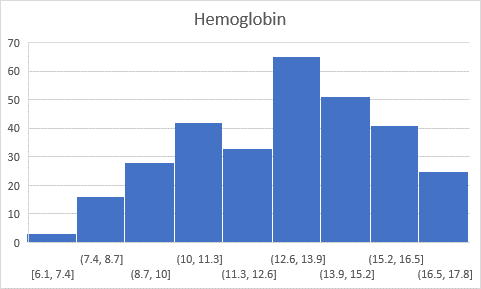


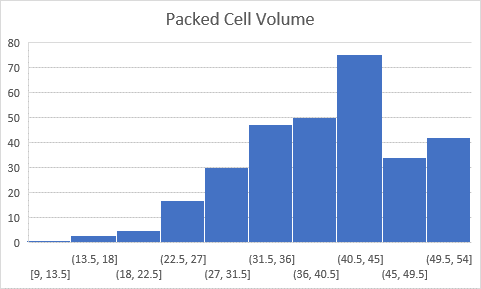


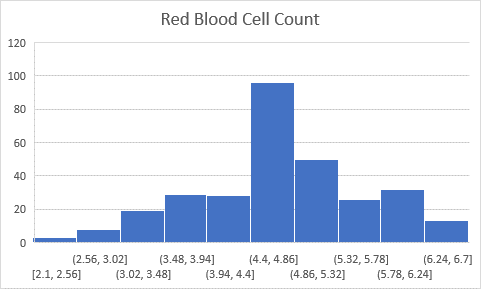


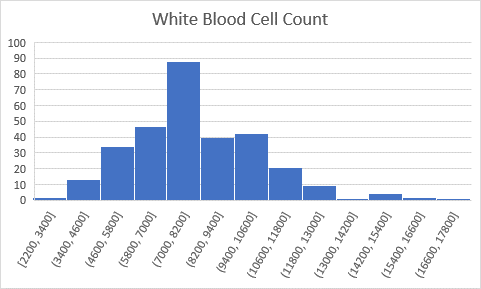


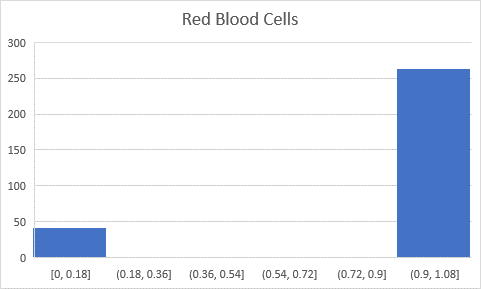


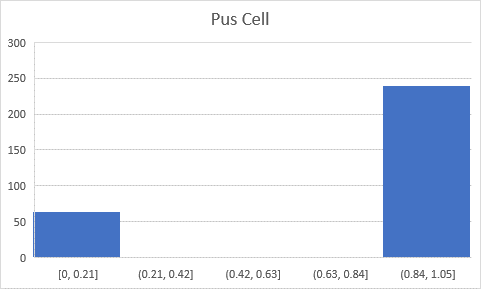


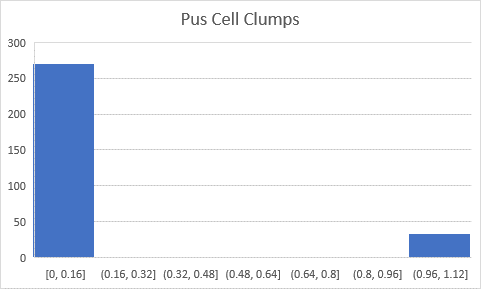


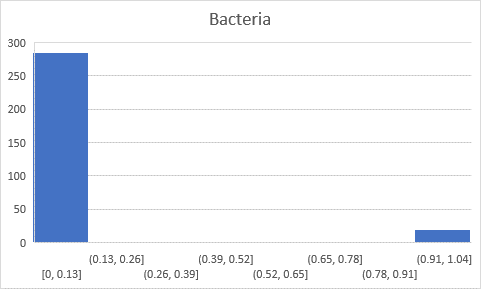


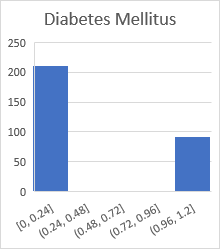
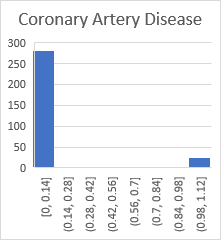


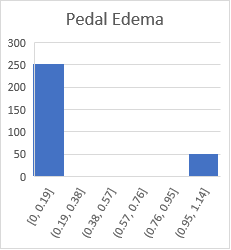
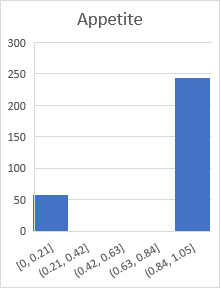


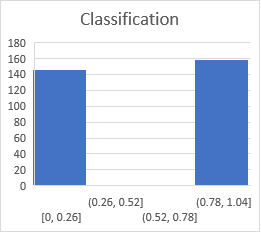
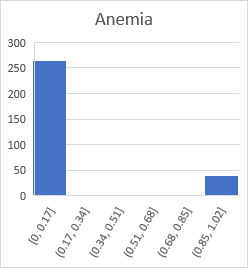






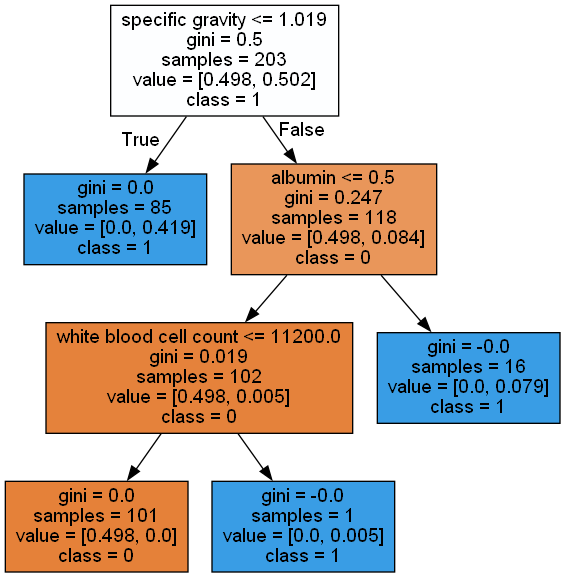
 

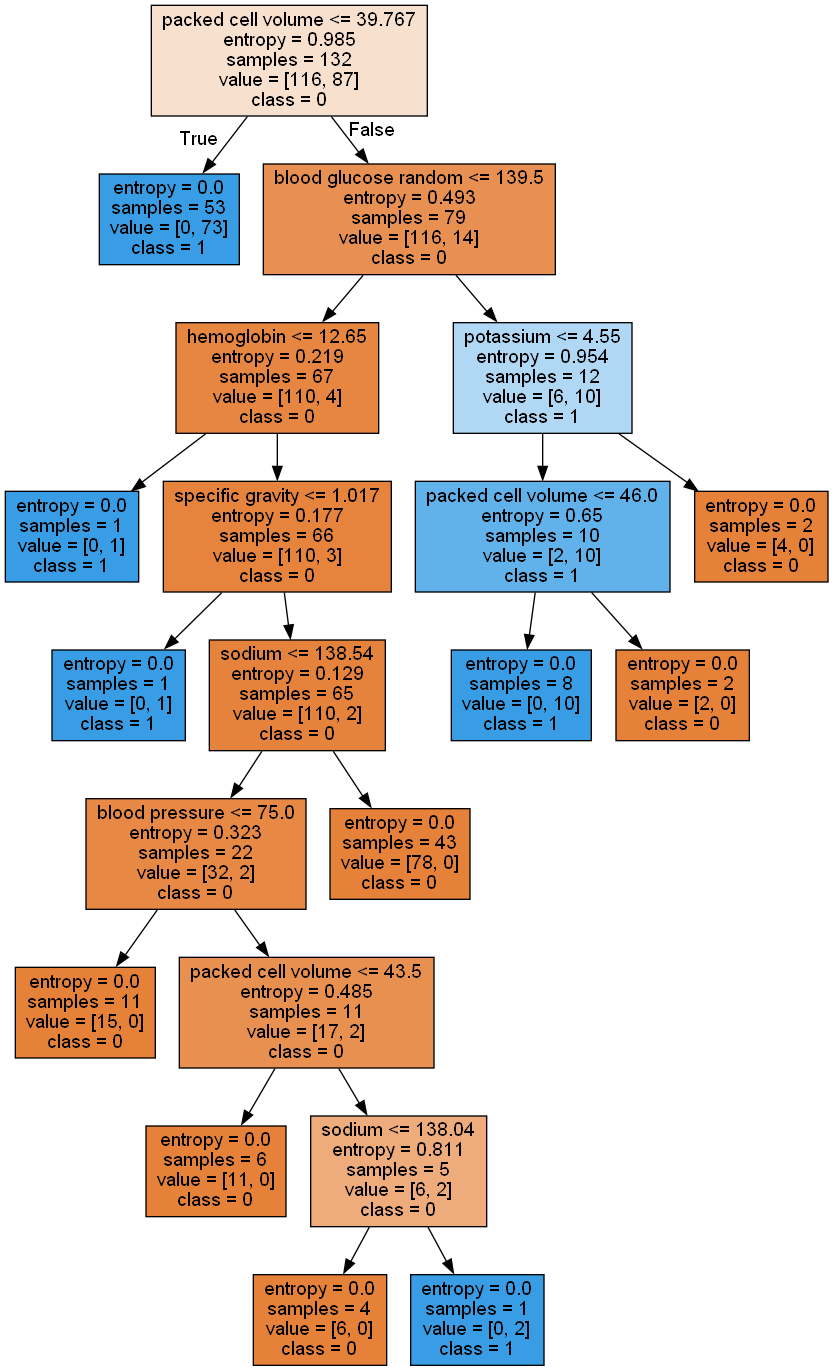




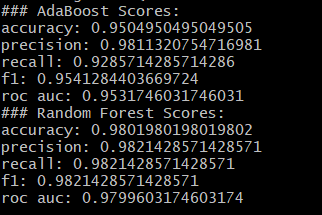
סיווג מבוסס עצים

1. **בחירת שיטות לסיווג נתונים:**כפי שהוזכר בשאלה 1 סעיף ד, בחרנו בשתי שיטות ensamble לצורכי סיווג הנתונים – AdaBoost וRandom Forest. שתי השיטות הללו נבחרו על פני שיטות סיווג שונות בגלל החוזק שבensamble. שיטות ensamble מייצרות מספר עצים, ואז נותנות לבחירת הרוב להכריע את הסיווג (בין אם רוב ממושקל או הוגן). היכולת הזו של הצבעה מונעת overfitting, ומשפרת את הביצועים של הסיווג.
2. **תיאור שיטות הסיווג:**
   1. שיטת AdaBoost:  
      מרכיבים עצים בשיטת CART – מנסים להקטין את מדד הgini בכל שלב ושלב, וגוזמים את העץ במידת הצורך. כל עץ מתאמן על חלק אחד של הנתונים (למשל בעזרת חלוקת k-fold), ולכן מתקבלים עצים שונים. כשמסיימים עם הרכבת העצים משנים את העצים בעזרת למידה מטעויות. נותנים משקל גדול יותר לנתונים שבהם טעינו, ובכך מאמנים את העצים להשתפר בחיזוי של הטעויות שלהם. כשמסיימים את הרכבת העצים נותנים ציונים לעצים לפי החוזק שלהם על מידע הבדיקה. בעת חיזוי נקבע החיזוי הכולל לפי שיקלול של {מה עץ מסוים חזה, כמה הוא בטוח בחיזוי זה ומה משקל העץ} על כל העצים.
   2. שיטת Random Forest:  
      מרכיבים עצים בשיטת ID3/C4.5 – מנסים להגדיל את רווח הידע על בסיס אנטרופיה, בין אם רווח ישיר או רווח יחסי, ומוסיפים צמתים בעץ בהתאם. כל עץ מתאמן על חלק אחד של נתונים (למשל בעזרת חלוקת k-fold), ורק על חלק מהמאפיינים (העמודות בטבלה). כך נוצרים עצים המבוססים על מאפיינים שונים בחיזוי שלהם. בעת חיזוי נקבע החיזוי הכולל לפי דעת הרוב של העצים. התלות בהרבה עצים המתבססים על מאפיינים שונים עוזר להסיר overfitting.
3. **תוצאות ונתונים:**

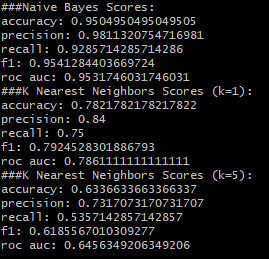
בתור שיטות ensamble, כל אחת מהשיטות מייצרת יער. בגלל שיער קשה להצגה, אציג עץ לדוגמה מכל אחת מהשיטות.  
שיטת Adaboost:  


שיטת Random Forest:ניתן לראות בעץ את בכל צומת הדברים הבאים:

* 1. תנאי – התנאי לפיצול של הצומת. אם התנאי מתקיים נפנה שמאלה, אם התנאי לא מתקיים נפנה ימינה. אם הוא מתקיים נפנה שמאלה. בעלים שורה זו לא קיימת. שורה זו גורמת לצורה של עץ בינארי.
  2. גודל המדד – בין אם משתמשים במדד אנטרופיה או ג'יני, אפשר לראות את הערך הנוכחי של המדד בהתאם למקום בעץ. ערך של 0 אומר שבהכרח הצומת הוא עלה.
  3. כמות הנתונים – כמות הנתונים מתוך הטבלה המקורית שמקיימים את כל התנאים להגעה לצומת הזה. ערך של 1 אומר שבהכרח הצומת הוא עלה.
  4. ערך – לא רלוונטי, שדה לשימוש פנימי של הספריות.
  5. זיהוי – הזיהוי המקומי בעבור הצומת הנוכחי – האם חולה בCKD או לא. ערך 1 (או צבע רקע כחול) זה חולה, וערך 0 (או צבע רקע כתום) זה לא חולה.

1. **חישובי דיוק:**חישבתי בעבור כל אחד מהעצים דיוקים במדדים שונים, בהתאם לחלוקה של אימון-בדיקה. התוצאות שהתקבלו הן התוצאות הבאות:  
   
2. **השוואה ומסקנות:**  
   ראשית, נתחיל מהשוואה לחוק הרוב. אם נלך לפי חוק הרוב, רוב הרשומות הן של חולים (classification=1), והן 250 רשומות מתוך 400. כלומר אם ננחש על אדם מסויים שהוא חולה לפי חוק הרוב נהיה צודקים ב62.5% מהמקרים. בבירור, שתי השיטות שלנו עברו את הביצועים הללו בקלות, ולכן אפשר להגדיר את כריית המידע כהצלחה.  
     
   שנית, נשווה בין שתי השיטות השונות. כמובן שבגלל שהן שיטות ensamble נקבל עליהן תוצאות טובות, אבל במקרה הזה לRandom Forest יש תוצאות טובות יותר בכל אחד מהמדדים. מכיוון שבמקרה כזה דיוק הניתוח חשוב בהרבה ממהירות הניתוח, ככל הנוגע לרפואה, כמובן שנעדיף את המודל שמביא תוצאות יותר טובות, גם במקרה שהוא איטי יותר. לכן, נבחר במודל Random Forest בתור המודל המועדף.  
     
   לבסוף, נבחן מה היה יכול להתבצע אחרת בשביל להוביל לתוצאות יותר טובות או אמינות. לדעתי, מסד הנתונים שקיבלנו הוא די קטן ומנוון, ומעבר לכך היו חסרים בו הרבה נתונים. זה הוביל לכמות קטנה של נתונים, ובגלל השלמת הנתונים הרבה מהשדות היו דומים בערכיהם. על כן, הצלחנו להגיע לתוצאות יחסית טובות בעזרת יערות יחסית קטנים. להבא, נכון להביא מסד נתונים גדול בהרבה, ועם ערכים יותר מדוייקים. זה יסיר מליבי את החשש למעין overfitting שיכול להיות שנוצר פה. בנוסף, ייתכן שהיו נתונים מדעיים נוספים שהיו יכולים לעזור לחיזוי, אך מפאת חוסר הידע הרפואי והמדעי שלי לא אדע להמליץ על נתונים כאלה להבא.

סיווג מבוסס בייס

1. **תיאור ובחירת שיטה:**
   1. שיטת Naïve Bays:  
      שיטה זו מניחה שהתכונות בלתי תלויות, ובכך מקלה משמעותית על החישובים הסטטיסטיים שיש לבצע. **יתרונות:** חישובים פשוטים וקצרים, זמן הרכבה וריצה מהיר יותר, אין צורך לנתח תלות בין תכונות לפני התחלת ריצת האלגוריתם.  
      **חסרונות:** חוסר דיוק במידה ואכן יש תלות בין התכונות.
   2. שיטת רשת בייסיאנית:  
      שיטה זו מתבצעת באמצעות רשת המחברת בין תכונות עם צלעות מכוונות. כל צלע מציינת תלות בין שני משתנים. הצלעות ברשת חד כיווניות.  
      **יתרונות:** מאפשר התייחסות סטטיסטית לקשר בין התכונות הנתונות, ולכן מציע דיוק טוב יותר במידה והרשת מוגדרת היטב.  
      **חסרונות:** חישוביות כבדה יותר, דורש לדעת את כל התלויות בין התכונות מראש.
   3. בחירה:  
      בעבור משימה זו נבחר בשיטת Naïve Bays. שיטה זו אומנם פחות מדוייקת מאשר רשת בייסיאנית במידה ומכירים את כל התלויות בין התכונות, אבל היתרונות המובהקים שלה קריטיים לצרכי המשימה. ראשית, בתור מומחי כריית מידע ולא מומחי מדעי החיים/רפואה, יהיה לנו קשה מאוד לקבוע את התלויות בין התכונות, דבר שעלול להוביל לתוצאות שגויות בשיטת הרשת הבייסיאנית. לכן, ההנחה של חוסר התלות בין התכונות מקלה משמעותית על תהליך הכנת הנתונים שלנו.שנית, שיטת Naïve Bays מאפשרת זמני אימון וריצה קצרים ביחס לרשת בייסיאנית, ובמקרים דחופים של סיווג הבדלי הזמנים עלולים להיות קריטיים.
2. **פתרון הבעיה בעזרת שימוש בK-NN:**
   1. הכנת נתונים: כפי שעשינו בתחילת המסמך.
   2. חלוקה לקבוצות: לחלק את הנתונים לקבוצת לימוד וקבוצת ביקורת. באופן דומה למודלים האחרים.
   3. בחירת הערך של k: כלומר מה מספר השכנים הקרובים שעליו נתבסס. ערכים שונים של k עלולים להוביל להבדלים גדולים בתוצאות הסיווג.
   4. בחירת מדד למרחק: כחלק מהאלגוריתם מחשבים מרחק בין נקודות רב מימדיות. סוג המרחק שמחשבים משפיע על תוצאות האלגוריתם.
   5. חישוב המרחקים: נחשב בעבור כל אחת מהנקודות בקבוצת הביקורת, את המרחק שלה לנקודות בקבוצת הלימוד.
   6. בחירת הרוב: בעבור כל אחת מהנקודות בקבוצת הביקורת, נסווג את הנקודה לפי הסיווג של k הנקודות הקרובות ביותר אליה (השכנות שלה).
   7. ניתוח תוצאות: השוואה סטטיסטית בין התוצאות שהתקבלו לבין התוצאות הרצויות.
   8. חזרה והתאמה: ניתן לחזור לשלב c או לשלב d ולשנות את הבחירה, ובכך לקבל תוצאות שונות ולבצע התאמות עד אשר המודל המתקבל אופטימלי.
3. **בחירת פתרון:**בעבור משימה זו נבחר בשיטת Naïve Bayes. בנוסף לכל היתרונות שתיארתי בסעיף א, מסד הנתונים שאיתנו אנחנו עובדים רווי במשתנים בינאריים וקטגוריים, ואלה משפיעים לרעה על סיווג מבוסס מרחק. בנוסף, יהיה קשה להגדיר פונקצית מרחק חד משמעית התאפשר לשלב בין כל התכונות הקיימות במסד הנתונים בגלל מגוון הסוגים והטווחים שלהם, וכן ההתאמות של הפונקציה יהיו גם הן קשות מאוד. בנוסף, בגלל חוסר המגוון בנתונים שקיבלנו, ייתכן ושימוש בKNN יצמיד שכנים "רחוקים" יחסית למה שאמורים לקחת, בגלל שהנקודות לא צפופות מספיק. זאת ניתן לבדוק בעתיד באמצעות ניתוח אשכולות.
4. **הרצה והצגת תוצאות:**  
   חישבתי בעבור כל אחד מהמודלים דיוקים במדדים שונים, בהתאם לחלוקה של אימון-בדיקה. התוצאות שהתקבלו הן התוצאות הבאות:  
   
5. **ניתוח תוצאות והסקת מסקנות:**שוב, נתחיל מהשוואה לחוק הרוב. אם נלך לפי חוק הרוב, רוב הרשומות הן של חולים (classification=1), והן 250 רשומות מתוך 400. כלומר אם ננחש על אדם מסויים שהוא חולה לפי חוק הרוב נהיה צודקים ב62.5% מהמקרים. לכן, גם במקרה הזה, שתי השיטות שלנו עברו את הביצועים הללו, ולכן אפשר להגדיר את כריית המידע כהצלחה.  
     
   שנית, נשווה בין שתי השיטות השונות. אמנם בחרנו באלגוריתם Naïve Bayes, אבל רציתי לבדוק האם השערתי נכונה ואכן נקבל בו תוצאות טובות יותר מאשר באלגוריתם KNN, גם בעבור kים שונים. אז בחרתי שני kים סטנדרטים, 1 ו-5, והרצתי את אלגוריתם KNN ביחד עם Naïve bayes. בבירור קל לראות שאכן אלגוריתם Naïve Bayes קיבל תוצאות טובות משמעותית מKNN, בהתאם למה שביססתי בבחירת הפתרון. בנוסף, הרצת KNN עם k=5 הניבה תוצאות טובות פחות מאשר k=1, שוב כפי שחשבתי וכתבתי בסעיף ג.  
     
   לבסוף, נבחן מה היה יכול להתבצע אחרת בשביל להוביל לתוצאות יותר טובות או אמינות. חשוב לציין כי אם היו ידועים לנו כל התלויות מראש, היינו יכולים להשתמש ברשת בייסיאנית במקום באלגוריתם בייס הנאיבי, וייתכן שהבדל זה היה מניב לנו תוצאות טובות יותר. בנוסף, גם באלגוריתם KNN היו התאמות שכנראה היינו יכולים לבצע בשביל לקבל תוצאה טובה יותר, למשל כפי שאמרתי הרכבת פונקציית מרחק ייעודית. עם זאת, דבר זה קשה אלגוריתמית וגם השיפור שלו יהיה קשה, ולכן אין זה היה ישים לפרוייקט זה, אך בהחלט ייתכן שיהיה ישים לפרוייקטים עתידיים.

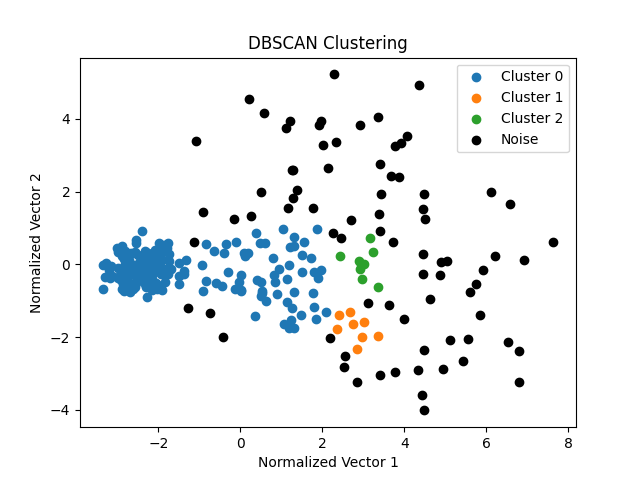
חלוקה לאשכולות

1. **מדדי איכות לאשכולות:**
   1. מינימום מרחק בתוך אשכול: נרצה שהמרחק בין שני האיברים הרחוקים ביותר בתוך האשכול יהיה קטן ככל הניתן, כלומר שכל אשכול יהיה צפוף מאוד.
   2. מקסימום מרחק בין אשכולות: נרצה שהמרחק בין שני האיברים הקרובים ביותר בין אשכולות שונים יהיה גדול ככל הניתן, כלומר האשכולות יהיו מרווחים אחד מהשני.
   3. גילוי תבניות נסתרות: אפשר למדוד כמה האיכות של החלוקה לאשכולות טובה לפי כמות התבניות שהתגלו – תבניות הזהות בתוך האשכול המתגלות על בסיס החלוקה לאשכולות, לפי מספר מצומצם של תכונות.
2. **גישה לניתוח אשכולות:**
   1. שיטת KMeans:  
      שיטה זו מגדירה מראש מספר אשכולות רצוי, ומגדירה באקראי מרכזים לאשכולות האלה, מבין הנתונים. משם האלגוריתם עובר איטרציות של חלוקה של האיברים לאשכולות המוגדרות כבר, והעברת איברים בין אשכולות עד שאף איבר לא עובר, ובכך הגענו למינימום מרחק בין נקודה למרכז האשכול שלה.  
      **יתרונות:** האלגוריתם פשוט למימוש וקל יחסית מבחינה חישובית, ואם ידוע מספר האשכולות הדרוש הוא נותן תוצאות מצויינות.  
      **חסרונות:** המוגבלות הגיאומטרית של האלגוריתם אומרת שהוא נותן רק צורות דומות למעגל רב מימדי, ולא מאפשר צורות כמו טבעות. בנוסף, אם לא יודעים את מספר האשכולות מראש האלגוריתם לא שמיש.
   2. שיטת אשכול היררכי:  
      שיטה זו מרכיבה את האשכולות שכבה שכבה. אפשר להתחיל מלמעלה למטה או מלמטה למעלה, אבל התוצאה תהיה זהה. בכל שכבה יש כמות אשכולות שקטנה ב1 מהשכבה שמתחתיה, כך ששתי האשכולות הכי קרובות בשכבה התחתונה ישתלבו בשכבה העליונה. שיטה זו נותנת את כל האופציות האופטימליות לכל כמות אשכולות.

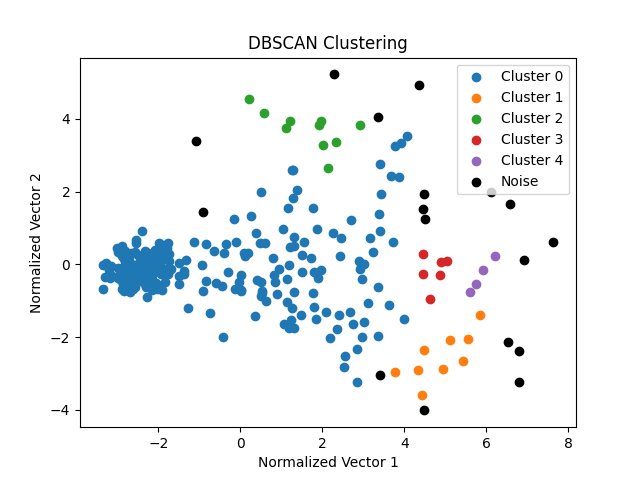
**יתרונות:** האלגוריתם "נותן בחינם" את הפתרון בעבור כל כמות אשכולות, באופן שמאפשר בחירה גמישה של כמות האשכולות הדרושה לאחר שרואים את התוצאות.   
**חסרונות:** החישובים הרבים של האלגוריתם הופכים אותו למסובך יותר, וגם לו יש מגבלה גיאומטרית כמו לKMeans. בנוסף, הוא דורש ידע רב מהמשתמש בשביל להכריע באיזו כמות אשכולות לבחור.

* 1. שיטת DBSCAN:  
     שיטה זו מגדירה צפיפות דרושה לקשר בין שתי נקודות במרחב הרב מימדי. לאחר מכן היא מחברת את כל הנקודות המקיימות את התנאים, עד שהיא מסיימת את האשכול הנוכחי, ועוברת כך להגדיר את האשכול הבא עד שכל הנקודות מוגדרות כחלק מהאשכולות או כרעש (במידה ואין מספיק נקודות המחוברות אליהן).   
     **יתרונות:** יותר רובסטי ומאפשר צורות מיוחדות של אשכולות שהשיטות האחרות לא יכולות להפיק, וכן מאפשר להבדיל בין נקודות רעש לנקודות מעניינות באשכולות.  
     **חסרונות:** יש לבצע הרבה התאמות למשתנים כמו המרחק המקסימלי והכמות המינימלית של נקודות לאשכול בשביל לקבל תוצאות מדוייקות, וכן שינוי קטן בפרמטרים עלול להוביל לתוצאה שונה לחלוטין. בנוסף, האלגוריתם לא מאוד יעיל בצורתו הבסיסית.
  2. בחירה:  
     בעבור משימה זו נבחר בשיטת DBSCAN. שיטה זו מאפשרת גמישות משמעותית על השיטות האחרות, וגם היכולת שלה לסנן נקודות רעש מאוד יעילה וחשובה למשימה. הבדלי זמן הריצה של האלגוריתם הזה למול שתי השיטות האחרות אמנם קיימים, אבל בעזרת אופטימיזציות ומבני נתונים מתאימים אפשר להקטין את ההפרש לזמן ריצה שאפשר לעבוד איתו. מציאת הפרמטרים האופטימליים גם היא בעיה, אבל היא דומה לבעיה בשיטה KMeans, ויותר קל לעשות אותה באופן אוטומטי מאפשר הבחירה בKMeans או בשיטות ההיררכיות.

1. **שלבי ניתוח האשכולות:**
   1. בחירת פרמטרים: בחירת המרחק המקסימלי, והכמות המינימלית של איברים באשכול.
   2. בחירת מדד למרחק: כחלק מהאלגוריתם מחשבים מרחק בין נקודות רב מימדיות. סוג המרחק שמחשבים משפיע על תוצאות האלגוריתם.
   3. מציאת נקודות ליבה: נמצא את הנקודות אשר יש להן לפחות כמות איברים מינימלית של נקודות "קרובות מספיק" (שהמרחק בינהן קטן מהמרחק המקסימלי). מנקודות הליבה נחבר נקודות בשלבים הבאים. נקודות הליבה הן נקודות שבהכרח חלק מאשכול.
   4. הרכבת אשכולות: נעבור לפי נקודות ליבה שלא ביקרנו בהן בעבר. לנקודה כזו שנמצא נגדיר אשכול חדש. כעת נעבור על כל הנקודות ה"קרובות מספיק" לנקודת הליבה. נוסיף כל אחת מהן לאשכול הנוכחי. נמשיך ונעבור על כל הנקודות ה"קרובות מספיק" של כל אחת מהנקודות שכבר באשכול, עד שלא נמצא נקודות חדשות להוסיף לאשכול. נחזור ונמשיך כך עד שעברנו על כל נקודות הליבה.
   5. מציאת נקודות רעש: כל הנקודות שבסוף שלב d לא חלק מאשכול כלשהו, הן נקודות רעש.
   6. ניתוח תוצאות: בדיקה האם האשכולות שמצאנו עומדות בתנאים שהגדרנו. במידה וכן, נקבל את התוצאה הנוכחית. במידה ולא, ניתן לבצע חזרה והתאמה.
   7. חזרה והתאמה: ניתן לחזור לשלב a או לשלב b ולשנות את הבחירה, ובכך לקבל תוצאות שונות ולבצע התאמות עד אשר מתקבלת החלוקה הדרושה.
2. **הרצה והצגת תוצאות:**  
   בעבור מרחק מקסימלי 0.5 ומספר מינימאלי באשכול 5 מתקבל הגרף:



בעבור מרחק מקסימלי 0.75 ומספר מינימאלי באשכול 4 מתקבל הגרף:

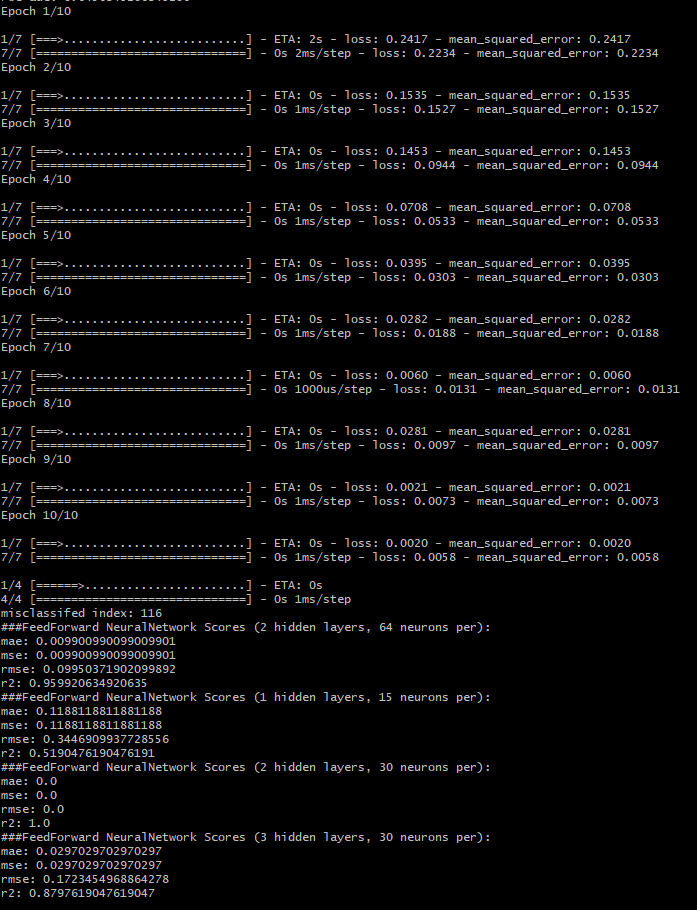


1. **ניתוח תוצאות והסקת מסקנות:**בשני המקרים, התקבל אשכול אחד גדול מאוד, ומספר קטן של אשכולות קטנים. זה בגלל הקרבה בין הנתונים, וכנראה גם באשמת הנירמול של הנתונים, שהתבצע לצורך הצגת התוצר. לעומת זאת, כן ניתן לראות פה בצורה ברורה את היתרון הגדול של DBSCAN, שהוא הצורות המסובכות שהוא מסוגל להניב. היה קשה לקבל תוצאה כזו מורכבת בעבור אלגוריתם אשכולות אחר.  
   לעומת זאת, עם כמות הרעש שהתקבלה בהרצה הראשונה של האלגוריתם, נראה שהיה צורך להחליף את הנתונים ולכן עברתי לסט הפרמטרים השני.  
   בסט הפרמטרים השני, התקבלה התאמה גבוהה מאוד בין הנבדקים הבריאים לאשכול הראשון, ואילו רק כמות קטנה של חולים נכנסו לאשכול הזה. כל האשכולות הקטנים יותר והרעש מייצגים חולים.

סיווג מבוסס רשת נוירונים מלאכותית

1. **ארכיטקטורת הרשת:**נרצה להשתמש ברשת נוירונים מלאכותית לסיווג הנבדק. לשם כך, נשתמש ברשת FeedForward מלאה. המשמעות של ארכיטקטורה זו היא שהרשת מורכבת משלושה סוגים של שכבות: שכבת קלט, שכבה חבויה, ושכבת פלט. השכבה הראשונה ברשת היא שכבת הקלט, שבה יש נוירונים מייצגים לקלטים שלנו מהטבלה. השכבה האחרונה היא שכבת הפלט, שבה יש נוירון אחד – שיהיה דלוק במידה והנבדק חולה, וכבוי במידה ואינו חולה. השכבות באמצע יהיו שכבות חבויות, שלהן מספר נוירונים שיקבע לאחר ניסוי וטעיה על כמויות שונות ותוצאות שהן מקנות. השכבות החבויות יקבלו קלט מהשכבה שלפניהן, כך שכל נוירון בשכבה X יהיה מקושר כקלט לכל הנוירונים בשכבה X+1. לאחר מכן, יינתן לכל קלט של הנוירון משקל, ויינתן לו היסט (bias). לפי פונקצית ההפעלה שנקבע – ReLU, הסכום של הקלטים והייסט בהתאם למשקלם יקבע האם הנוירון יודלק, ואם כן איזה קלט הוא יוציא. ReLU מקבלת כקלט מספר רציונלי, ומחזירה 0 אם הוא שלילי, או את הקלט אחרת. פונקציית ההפעלה הזו מאפשרת תוצאות טובות מכיוון שהיא אינה ליניארית, כלומר אין צמצום מתמטי של השכבות השונות לכדי שכבה אחת, ולכן מתאפשרת למידה טובה יותר, ומקבלים את היתרונות של רשת מרובת שכבות. בשכבת הפלט נשתמש בפונקצית האקטיבציה Sigmoid, שתאפשר סיווג טוב יותר לתוצאה בינארית, שזה מה שאנחנו מחפשים. לאחר ניסוי וטעייה, נראה שהוספת שכבות חבויות או הגדלת גודל השכבה עוזרת לשפר את תוצאות הרשת בעת האימון, כנראה כתוצאה של התאמת יתר לנתוני האימון. לכן הוספתי מספר ניתוחים שונים של רשתות בארכיטקטורות שונות (כי על מספר נתונים כזה זמן האימון קצר יחסית אז אפשר לנסות יותר). אם הייתי צריך לבחור אחת מהאופציות, הייתי בוחר ברשת שבה 2 שכבות של 64 נוירונים, שכן היא הגיעה לתוצאות טובות בלי להיות מורכבת מדי. עוד על המסקנות על ארכיטקטורת הרשת בסעיף ה (ניתוח תוצאות והסקת מסקנות).
2. **הפרמטרים בעבור האופטימיזציה:**
   1. פונקציית השגיאה: פונקצית השגיאה היא הפונקציה לפיה הרשת מחשבת את השגיאה שלה בחיזויים, ובכך בעזרת backpropagation משפרת את המשקלים ברשת לצורך צמצום הטעויות. פונקצית השגיאה שבה בחרתי היא mean squared error, כלומר הפרש ריבועים ממוצע. בחרתי בפונקציה זו בגלל כמות הנתונים המספריים שיש לנו, ובגלל שבעזרתה אפשר לשלב גם נתונים בינאריים בלמידה (בעזרת הפיכתם למספרים).
   2. גודל הbatch: גודל הbatch הוא פרמטר המגדיר כמה פיסות נתונים (שורות במסד הנתונים) יעברו בבת אחת ברשת, ולפיכך תותאם הרשת לפי התוצאות של כל פיסות הנתונים הנ"ל. לאחר ניסוי וטעייה, נראה שהפרמטר הזה לא משפיע יותר מדי על תהליך הלמידה ברשת שהגדרתי, ולכן בחרתי בו להיות הערך "ברירת המחדל" שמצאתי באינטרנט, שהוא 32.
   3. קצב הלמידה (learning rate): קצב הלמידה מגדיר כמה מהר המודל משתנה מהנתונים. כלומר, בכמה אחוזים המודל ישתנה על הbatch הנוכחי של הנתונים, כמובן בשביל להשתפר לזהות את הנתונים הללו בצורה יותר טובה. בפרוייקט הזה בחרתי בקצב הלמידה שמגיע בתור ברירת מחדל באופטימזטור שלקחתי בתהליך הלמידה, שנקרא Adam. הערך המספרי של קצב הלמידה הוא 0.1%. יש לציין שניסיונות לשנות את המספר הזה לא השפיעו במיוחד על תהליך הלמידה, ולכן השארתי אותו כך.
   4. כמות החזרות (epoch): כמות החזרות מגדירה כמה ריצות על כל הנתונים יתקיימו לפני שהרשת תוגדר כמוכנה. בכל epoch עוברים כל הנתונים דרך הרשת. בפרוייקט הזה בחרתי בכמות חזרות להיות 10, מכיוון שהגדלת הepoch לא הניבה שיפור טוב בביצועים אחרי 10 בהשוואה לשיפור לפני 10.
3. **הרצת הרשת וביצועיה:**

תוצאות ההרצה: (צירפתי גם את תהליך ההרכבה של הרשת רק בעבור אחת מהרשתות).



פונקצית ההפסד ביחס למספר הepoch:

1. **מקרים חריגים:**  
   לאחר ההמרה של התוצאות לבינאריים, עלו מבין ההרצות הרבות שלי רק שתי שורות שבהן הרשת מזהה באופן שגוי, והן id 116, id 75. בעבור נבדק 116, ההשערה שלי על הסיווג השגוי היא שחסרים לו הרבה נתונים, ופרט לנתונים שחסרים הבדיקות היחידות שבהן הוא יוצא מן הכלל הן תאי PUS והעובדה שהוא חולה באנמיה. בגלל שהחלפתי סימני שאלה בתוצאות ממוצעות, כנראה החישוב הכולל של הסיכוי שיהיה חולה מוטה לרעה, ולכן מנחשים שהוא בריא למרות שהוא חולה. בעבור נבדק 75, לא חסרים לו שום נתונים, וכל התוצאות שלו נראות סבירות פרט לתאי PUS ולתאי הדם האדומים. במקרה זה ייתכנו שתי אופציות – או שהוא חולה ופשוט אין לו מחלות רקע או בדיקות רקע שמעידות על כך, או שהוא בריא והייתה טעות במסד הנתונים ההתחלתי על הסיווג שלו. בכל מקרה, בגלל ששתי אלה השגיאות היחידות, ניתן להגדיר את כריית המידע כהצלחה.
2. **ניתוח תוצאות והסקת מסקנות:**נחשב את הmae ואת הrmse של כלל הרוב: (על מסד הנתונים המפולטר הפעם, שבו 146 בריאים ו158 חולים)  
   בגלל שכלל הרוב היה טועה ב146 מקרים מתוך 304, מתקבל ישירות שהMAE של כלל הרוב היה .  
   בגלל שבסיווג בינארי הMSE והMAE זהים, היה מתקבל לכלל הרוב RMSE שהוא: .

התוצאות של הרשת הרבה יותר טובות מתוצאות אלה, שכן במדדי שגיאה שואפים לשגיאה נמוכה יותר. יש לציין שהרשת הגיעה לביצועים מאוד טובים, שבמקסימום יש לה שתי טעויות, במקרה הממוצע טעות אחת ובמקרה הטוב אין טעויות בכלל (כמות הטעויות נובעת מהאם נבדק 116 ונבדק 75 נבחרים לקבוצת האימון או הבדיקה, שכן טעויות בהם עולות רק בקבוצת הבדיקה).  
תוצאות אלה עלולות להעיד על התאמת יתר, אך העובדה שהייתה הפרדה מראש בין נתוני בדיקה לנתוני אימון מוכיחה שקשר זה אינו התאמת יתר של הרשת לנתונים, אלא אוליי דמיון יתר בין הנתונים, או פשוט "סיווג טוב". בכל מקרה התוצאות שהתקבלו ברשת טובות מאוד, וכריית המידע באמצעות רשת נוירונים מלאכותית התבצעו בהצלחה.

סיכום ומסקנות

בתחילת המסמך מוצגים התוצרים ממטלה מס' 21, שבה ביצענו את הכנת הנתונים לכריית המידע, והרכבנו עצי סיווג, שבהם המטרה היא למצוא מאפיינים המבדילים בין מקרים שונים במסד הנתונים בעזרת תכונה מסויימת. בהמשך המסמך מוצגים התוצרים ממטלה מס' 22, שבה חקרנו שיטות כריית מידע שונות, שמאפשרות סיווג באופן שונה מעצי סיווג, ובינהן סיווג מבוסס בייס (סטטיסטיקה), סיווג לפי שכנים, רשתות נוירונים מלאכותיות, וגם חלוקה של נתונים לקבוצות (באופן שגם יכול לשמש לסיווג אבל גם לתגלית תבניות בנתונים), כלומר חלוקה לאשכולות.

מבחינה מספרית, מודלי הסיווג שמצאנו מסווגים טוב יותר או פחות, והשיטות (מסודרות מהטובה ביותר להכי פחות טובה) מתוארות כאן:

1. רשת נוירונים מלאכותיות FastForward NeuralNetworks
2. יער סיווג Random Forest (InformationGain)
3. יער סיווג AdaBoost (Gini) וגם Naïve Bayes
4. שכנים KNN (k=1)
5. שכנים KNN (k=5)

הערה חשובה לגבי ההשוואה: AdaBoost ו Naïve Bayesקיבלו תוצאות זהות, ולכן ממוקמות באותה הדרגה, ולמרות שהמדדים שעליהם בדקנו את רוב האלגוריתמים למול המדדים שבהם השתמשנו ברשתות נוירונים, ברור כי רשת הנוירונים עבדה טוב יותר משאר אלגוריתמי הסיווג מכיוון שהייתה לה רק טעות אחת בממוצע, למול האלגוריתמים האחרים שלהם הייתה לפחות טעות אחת בוודאות (כי אף אחד לא קיבל תוצאה מושלמת בסיווג), ולכן לכל היותר טובים כמו הרשת המלאכותית.

החלוקה של הנתונים לאשכולות נתנה לנו תמונת מצב לגבי הקרבה של הנתונים שקיבלנו. בהחלט יש קבוצה גדולה מאוד של נתונים שמרוכזים באותה הסביבה, והם ברובם הנבדקים שאינם חולים בCKD. על כן, אפשר להניח אחד משני דברים – או שהיינו אמורים לקבל גם אשכול גדול לחולים, ובגלל מחסור בנתונים לא קיבלנו, או שהגיוני שיהיו כמה אשכולים קטנים לחולים, מכיוון שיש מגוון של דברים העלולים להוביל למחלה, אבל נוצר הרבה נקודות רעש מכיוון שלא היו מספיק תוצאות של חולים אשר קרובים בנתונים אחד לשני. אני נוטה להאמין לשני, בגלל הפאן הרפואי של הבעיה, והעובדה שהרבה תופעות בעולם הרפואה לא מוסברות לחלוטין. אם היינו מדרגים את אלגוריתם החלוקה לאשכולות DBSCAN בתור אלגוריתם סיווג, כנראה הוא היה מקבל את המקום הרביעי במקום 1NN, וזאת כיוון שהוא זיהה טוב את החולים ברעש ובאשכולות קטנים, אבל נתקל בקצת בעיה באשכול הגדול, בו את כל הבריאים אבל גם ריכוז בלתי מבוטל של חולים.

בנוסף, בגלל שהיה חסר דמיון בין הנתונים של החולים, ייתכן שבעבור מסד נתונים גדול יותר או משובח יותר, היינו מקבלים ציון טוב יותר למסווגים המניחים דמיון בין הערכים (שכנים וNaïve Network), ותוצאה פחות טובה לNaïve Bayes אשר מניח אי תלות בין המשתנים השונים. בוודאי שאם זה היה המקרה היינו מקבלים חלוקה יותר יפה ומדויקת לאשכולות של חולים.

מהפרוייקט הזה אפשר ללמוד הרבה על פרוייקטי כריית מידע בסדר גודל כזה. כלומר, עם מסד נתונים בגודל דומה. באופן מפתיע, האלגוריתם שההרכבה שלו לוקח הכי מעט זמן, וגם הריצה שלו מאוד מהירה ביחד לאלגוריתמים אחרים, נותן כמעט את התוצאות הכי טובות – יער סיווג. האלגוריתם שנתן את התוצאות הטובות ביותר הוא האלגוריתם שלוקח הכי הרבה זמן להרכיב ולהריץ (רשת נוירונים מלאכותית), ועל כן אם יש לבחור אחד מהם אפשר להתלבט עם עדיפות "התשובה הנכונה" על פני זמן ההכנה והריצה. מאידך גיסא, האלגוריתם שלוקח כנראה הכי הרבה זמן ריצה בעת סיווג (שכנים), נתן דווקא הוא את התוצאות הטובות פחות, כנראה כאמור בגלל חוסר התאימות בין הנתונים של החולים. ברור שיש להגביל ולומר שהתוצאות היו יכולות להיות שונות בעבור מסדי נתונים יותר גדולים ופורים, ולכן במקרה כזה כדאי לנתח את ההתאמה מחדש.

ביבליוגרפיה

מקור בסיס הנתונים והמידע על הנתונים: <https://archive.ics.uci.edu/ml/datasets/chronic_kidney_disease>

ספריות פייתון שבהן השתמשתי:

Sklearn: <https://scikit-learn.org/stable/>

Graphviz: <https://graphviz.org/>

Pandas: <https://pandas.pydata.org/>

Numpy: <https://numpy.org/>

TensorFlow: <https://www.tensorflow.org/>

Keras: <https://keras.io/>