Data Mining ASSIGNMENT 21

Eli Ackerman

1. הגדרת הבעיה והכנת הנתונים

א. מטרת כריית המידע

מטרת כריית המידע היא לחזות מחלה כלייתית כרונית בנבדק נתון מתוך סט התכונות הנתונות לנו במאגר נתונים בעל 400 רשומות, כאשר תכונת class היא העמודה עליה נבצע את החיזוי (ckd or notckd).

הנחות:

נניח בפרויקט שכל הכנת נתונים שבוצעה ע"י כותבי המאמר בוצעה באופן מהימן

ב. הגדרת הנתונים

נגדיר כעת את הנתונים הגולמיים בהם נשתמש, להלן התכונות שנאספו על הנבדקים, נעזרתי במאמר ובפייתון:

שם תכונה	תיאור תכונה	סוג הנתונים	יחידות מידה	תחומי ערכים	ממוצע	סטיית תקן	נתונים חסרים
age	גיל	נומרי	שנים	>= 0	51.483	17.169	כן
bp	Blood pressure	נומרי	mm/Hg	>= 0	76.469	13.683	כן
sg	Specific Gravity	נומינלי	-	(1.005,1.010,1.015,1.020,1.025)	-	-	כן
al	Albumin	נומינלי	-	(0,1,2,3,4,5)	-	-	כן
su	Sugar	נומינלי	-	(0,1,2,3,4,5)	-	-	כן
rbc	Red Blood Cells	נומינלי	-	normal, abnormal	-	-	כן
pc	Pus Cell	נומינלי	-	normal, abnormal	-	-	כן
pcc	Pus Cell clumps	נומינלי	-	present, notpresent	-	-	כן
ba	Bacteria	נומינלי	-	present, notpresent	-	-	כן
bgr	Blood Glucose Random	נומרי	mgs/dl	>= 0	148.036	79.281	כן
bu	Blood Urea	נומרי	mgs/dl	>= 0	57.425	50.503	כן
sc	Serum Creatinine	נומרי	mgs/dl	>= 0	3.072	5.741	כן
sod	Sodium	נומרי	mEq/L	>= 0	137.528	10.408	כן
pot	Potassium	נומרי	mEq/L	>= 0	4.627	3.193	כן
hemo	Hemoglobin	נומרי	gms	>= 0	12.526	2.912	כן
pcv	Packed Cell Volume	נומרי	לא צוין	>= 0	38.884	8.990	כן
wc	White Blood Cell Count	נומרי	cells/cumm	>= 0	8406.122	2944.474	כן
rc	Red Blood Cell Count	נומרי	millions/cmm	>= 0	4.707	1.025	כן
htn	Hypertension	נומינלי	-	yes, no	-	-	כן
dm	Diabetes Mellitus	נומינלי	-	yes, no	-	-	כן
cad	Coronary Artery Disease	נומינלי	-	yes, no	-	-	כן
appet	Appetite	נומינלי	-	good, poor	-	-	כן
pe	Pedal Edema	נומינלי	-	yes, no	-	-	כן
ane	Anemia	נומינלי	-	yes, no	-	-	כן

ג. שלבי ה KDD

1. **הגדרת מטרות כריית המידע:** מטרת כריית המידע היא לחזות מחלה כלייתית כרונית בנבדק נתון מתוך סט התכונות הנתונות לנו במאגר נתונים בעל 400 רשומות, כאשר תכונת class היא העמודה עליה נבצע את החיזוי (ckd or notckd).

2. **איסוף ושמירת הנתונים**: הנתונים לכרייה הגיעו מרוכזים מהכתובת https://archive.ics.uci.edu/ml/datasets/chronic_kidney_disease
בנוסף הנתונים הועברו מפורמט arff לפורמט csv (באמצעות python)

3. **ניקוי הנתונים:** הנתונים שהתקבלו הכילו ערכים חסרים ולא נמצאו ערכים לא חוקיים, כמו כן לא נמצאו שורות כפולות, נפרט איך מלאנו את הערכים החסרים בהמשך.

4. **רדוקציה וטרנספורמציות על הנתונים:** בדיקה האם יש צורך ברדוקציה ע"י סינון רשומות ותכונות שלא רלוונטיות לתהליך הכרייה שלנו. בנוסף, ביצוע טרנספורמציה במידת הצורך באמצעות טכניקות כמו נרמול, דיסקרטיזציה ויצירת תכונות חדשות.

5. **בחירת שיטות וכלים לכריית מידע:** python for data preprocessing and prediction

6. **התאמת ייצוג הנתונים לשיטות כריית המידע שנבחרו:** חלק משיטות כריית המידע משתיתות אילוצים על אופן הצגת הנתונים, נבצע טרנספורמציות ודיסקרטיזציה נוספות במידת הצורך בכדי להתאים את הנתונים לאלגוריתמים.

7. **הרצת שיטות לכריית מידע שנבחרו:** נבצע את שיטות כריית המידע שנקבעו באמצעות python.

8. **ניתוח התוצאות:** בדיקת הממצאים מריצת האלגוריתמים וביצוע הערכה לפי מידת דיוק, רלוונטיות, פשטות וכו' ע"י ניתוח נתונים סטטיסטיים של ריצת האלגוריתמים.
לאחר ניתוח התוצאות קיימות שני אפשרויות:
   - התוצאות לא משביעות רצון – ניתן במצב זה להשתמש ב validation set על מנת למצוא את הפרמטרים שמשיגים את התוצאות הטובות לפני שנפעיל את המודל על ה test set .
   - התוצאות משביעות רצון – ניתן להתקדם לשלב של הסקת מסקנות

9. **הסקת מסקנות:** שימוש במודל שהתקבל בכדי לחזות את ה-test set או כל רשומה חדשה ללא label.
ניתן להציג את המודל בצורה ויזואלית, נוסחה מתמטית, כללי היסק, עץ החלטה, תלוי בשיטה שנבחרה.

ד. סקירה השוואתית בין החלופות האפשריות לביצוע כריית המידע

1. **Logistic Regression**

רגרסיה לוגיסטית היא אלגוריתם סיווג לינארי פשוט אך אפקטיבי. זה עובד על ידי הערכת ההסתברות שקלט שייך למחלקה מסוימת. במקרה שלנו, זה יעריך את ההסתברות שלמטופל יש CKD או לא.
עבור רגרסיה לוגיסטית, חשוב לבצע feature scaling, מכיוון שהוא רגיש לקנה המידה של תכונות הקלט.
Dimensionality reduction עשויה להועיל גם להפחתת הסיכון של overfitting.
יתרונות – זה פשוט וקל ליישום, זה יכול לעבוד היטב עם מספר רב של פיצ'רים, זה מספק הסתברות ברורה לסיווג.
חסרונות – הוא מניח קשר לינארי בין משתני הקלט לבין הפלט, ייתכן שזה לא יעבוד טוב אם הקשר בין הפיצ'רים לליבלים מורכב.

2. **עץ החלטה C4.5 עם Gain Raito**

C4.5 הוא אלגוריתם עץ החלטה המשמש הן לבעיות סיווג והן לבעיות רגרסיה. הוא עובד על ידי חלוקה רקורסיבית של הנתונים על סמך התכונה שמספקת את ה – gain ratio הרב ביותר. gain ratio דומה ל-information gain, אך הוא לוקח בחשבון גם את מספר הקטגוריות בתכונה.
יתרונות – יכול להתמודד עם פיצ'רים קטגוריים ורציפים, קל לפירוש ויכול לספק תובנות לגבי תהליך קבלת ההחלטות, יכול להתמודד עם noisy data.
חסרונות – יכול להיות רגיש לdata ועלול להתאים יתר על המידה (overfit), ייתכן שבממדים גדולים (number of data features) הוא לא יעבוד כל כך טוב.
אלגוריתם זה יודע להתמודד עם פיצ'רים רציפים אך עדיין דיסקרטיזציה יכולה להיות שימושית בפועל לשיפור הביצועים של המודל, בעזרת דיסקרטיזציה לפיצ'רים רציפים המודל יכול, אולי, לזהות נקודות פיצול משמעותיות ולשפר את מבנה העץ, בנוסף דיסקרטיזציה יכולה להוריד את המשקל של חריגים.
dimensionality reduction ו – feature selection יכולה לשפר את ביצועי המודל בצורה משמעותית.

3. **עץ החלטה CART עם Gini Index**

אלגוריתם עץ החלטות שפועל על ידי חלוקת הנתונים לקבוצות משנה קטנות יותר בהתבסס על Gini index.
מדד ג'יני מודד את ה-impurity של קבוצה של דגימות, והפיצול נבחר כך שלתת-הקבוצות המתקבלות תהיה הטומאה הנמוכה ביותר האפשרית. CART משמש הן לבעיות סיווג והן לבעיות רגרסיה.
יתרונות – קל לפירוש ומספק כללי החלטה ברורים, יכול להתמודד עם פיצ'רים קטגוריים ורציפים, יכול להתמודד עם noisy data.
חסרונות - יכול להיות רגיש לdata ועלול להתאים יתר על המידה (overfit), ייתכן שבממדים גדולים (number of data features) הוא לא יעבוד כל כך טוב.
תיאור הכנת הנתונים עבור אלגוריתם זה זהה ל – C4.5.

4. **Random Forest עם Gini Index (default)**

Random Forest הוא אלגוריתם למידה אנסמבל המשלב עצי החלטה מרובים כדי לשפר את הדיוק של התחזיות. האלגוריתם פועל על ידי יצירת עצי החלטה מרובים תוך שימוש בקבוצות משנה אקראיות של נתוני האימון והתכונות. התחזית הסופית מבוססת על ממוצע התחזיות של כל עצי ההחלטה ביער.
יתרונות – יכול להתמודד עם תכונות קטגוריות ורציפות כאחד, יכול להתמודד עם נתונים בממד גבוה, יש פחות סבירות להתאמת יתר מאשר עץ החלטה יחיד בגלל השילוב של מספר עצי החלטה.
חסרונות – יכול להיות יקר מבחינה חישובית והאימון ימשך זמן רב יותר, פירוש התוצאות יכול להיות מאתגר, במיוחד אם מספר העצים הוא גבוה במיוחד.

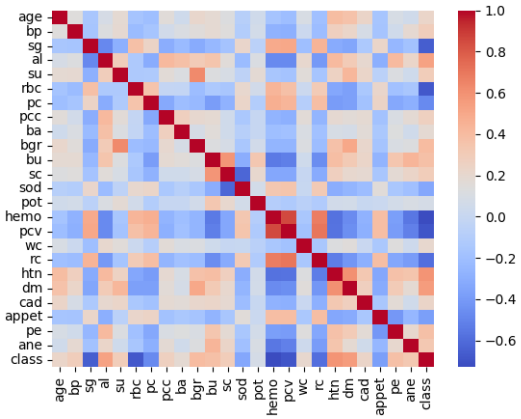
ה. שלבי הכנת הנתונים

נתונים חסרים

החלטתי למלא את הנתונים החסרים בתכונות הנומריות בממוצע של העמודה וזאת בגלל שזו היא שיטה שכיחה למלא תכונות נומריות אבל במיוחד למקרה שלנו, עבור כל תכונה נומרית רציפה לא מצאתי ערכים חריגים וכולם התפלגו דיי נורמלית, כמו כן הממוצע דיי חסין לחריגים.
בנוגע לנתונים חסרים בתכונות הנומינליות החלטתי למלא על סמך most frequent / hard voting / majority voting וזאת כי זאת שיטה שכיחה למילוי תכונות נומינליות וכן סביר מאוד להניח (סטטיסטית) שאם חסר איזשהו ערך, אזי מה שחסר הוא הערך שמופיע הכי הרבה.

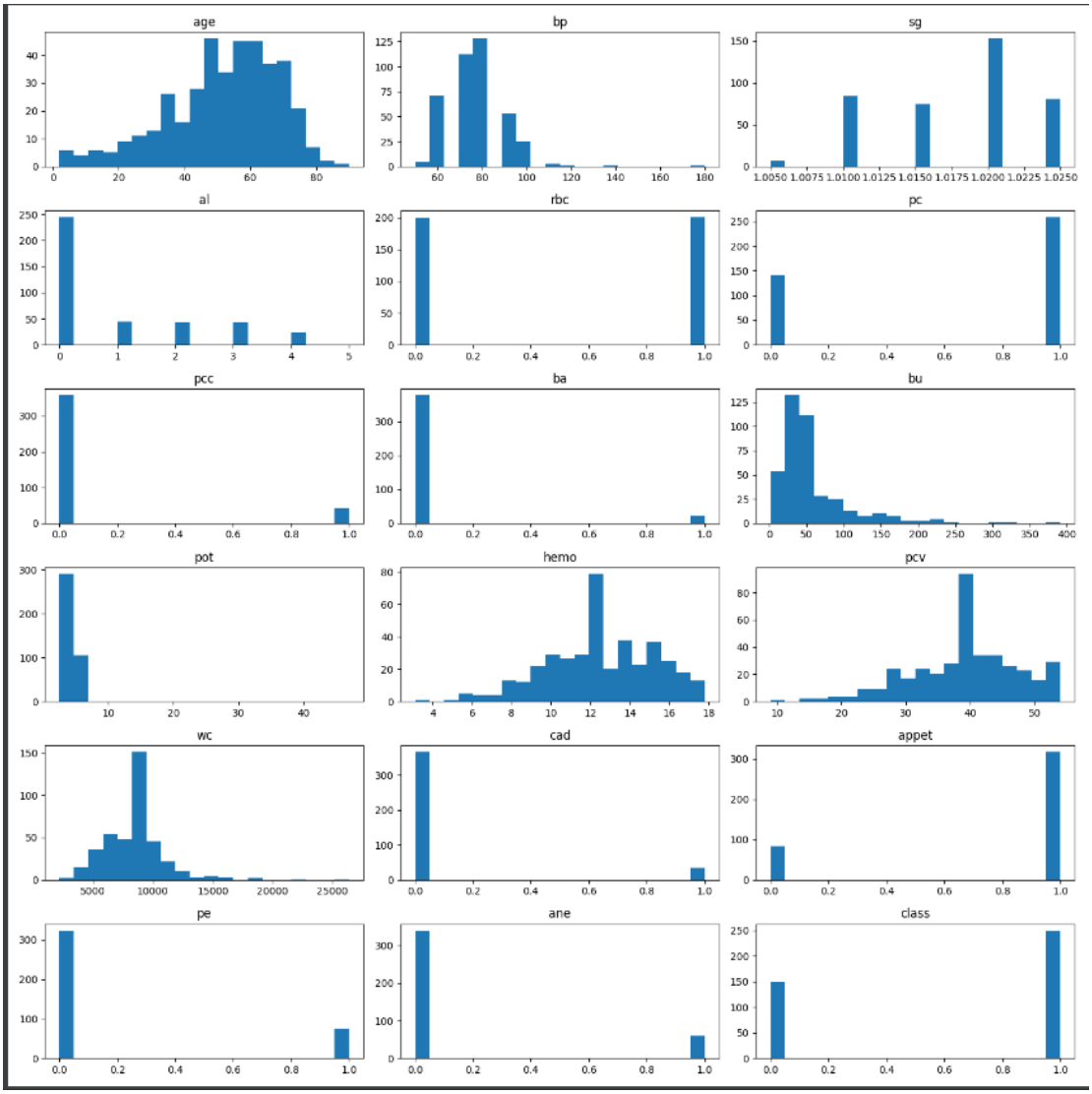
ניקוי נתונים

תחילה אנו נרצה לבדוק את רמת הקורלציה בין התכונות ובמידת הצורך למחוק לגמרי תכונות שלא רלוונטיות (תכונות שלא רלוונטיות לבעיית החיזוי שלנו או תכונות שהקורלציה שלהם עם תכונות אחרות גבוהה מאוד)
הוחלט שנמחק תכונות שהם highly correlated עם תכונות אחרות שהן לא תכונת החיזוי, וזאת כי אנחנו רוצים להוריד את ממד הבעיה אך לשמר כמה שיותר אינפורמציה רלוונטית, אנחנו לא רוצים רק להוריד כמה שאפשר ולאבד אינפורמציה חשובה, ה – threshold יהיה 0.6 (יחסית גבוה), להלן ה – correlation matrix:



חשוב לזכור ש - תכונות שהן highly correlated עם תכונת החיזוי נרצה שיישארו שכן הם משפיעים הכי הרבה על החיזוי (נבחר באותו threshold)
התכונות שהוסרו הן : {'sod', 'sc', 'rc', 'bgr', 'htn', 'su', 'dm'}

הצגת הנתונים



טרנספורמציה של הנתונים

החלטתי להפוך כל תכונה נומינלית עם שתי אופציות (למשל yes/no) לתכונה בינארית (0 או 1) וזאת כי חלק מהמודלים עובדים טוב יותר עם מספרים, בנוסף זה יותר נוח לניתוח (בתנאי שהחוקר מבין את המשמעות כמובן, אני מבין) ועוד יתרונות כמו למשל עזרה לstandardization, אפשרות ליצירת correlation matrix וכו'.

דיסקרטיזציה –
עבור הבעיה שלנו בחרתי ליישם את KBinsDiscretizier מהספרייה sckit-learn כדי להסוות את התכונות המספריות הלא-בינאריות ב-dataset שלנו.
בחרתי להשתמש ב KBinsDiscretizer -מכיוון שזו טכניקת דיסקרטיזציה בשימוש נרחב ומאפשרת לנו לציין את מספר הפחים שאנו רוצים ליצור. בחרנו להגדיר את מספר הפחים ל-8 כי אנחנו רוצים לקבל מספר סביר של קטגוריות תוך שמירה על כמות טובה של מידע עבור כל תכונה. לדוגמה, אם ניקח בחשבון את תכונת 'age', ונניח כי הערך המינימלי ב-dataset שלנו הוא 4 והערך המקסימלי הוא 90. עם 8 פחים, ניצור מרווחים של 11 שנים, כאשר הפח הראשון מייצג גילאים בין 4 ל-14, הפח השני בין 14 ל-25, וכן הלאה .כך נוכל לשמור על כמות טובה של מידע תוך הפחתת מספר הערכים.

2. סיווג

א. בחירת שיטות הסיווג

בחרתי במודלים הבאים: Random forest, CART .
סיבות לבחירתי:
Random Forest and CART הם שניהם מודלים מבוססי עץ החלטות, שיכולים להתמודד עם תערובת של נתונים נומינליים ונומריים. ב – dataset שלנו, יש את שני סוגי הנתונים, כך שהמודלים האלה מתאימים.
בנוסף, שניהם ידועים בפרשנות שלהם, עצי החלטה ניתנים להבנה בקלות ע"י בעלי עניין שלא בהכרח מבינים את הצד הטכני של המודלים, במקרה שלנו מדובר ברופאים ומטופלים שעשויים לרצות להבין כיצד המודל הגיע לתחזיות שלו.
Random Forest הוא מאוד יעיל וטוב כאשר יש רעש או שיש מספר רב של תכונות, כמו כן מודלים אלה הוכחו כיעילים בחיזוי תוצאות רפואיות שונות (במאמר Random Forest השיג את התוצאות הטובות ביותר), כך שיש רשת ביטחון.

ב. תיאור שלבי השיטות

CART:

האלגוריתם מתחיל בשורש העץ, עם סט הנתונים המלא.
בכל שלב מחשבים את הרווח האינפורמטיבי ואת מדד הפיצול של כל תכונה, מחפשים את התכונה שתביא לרווח אינפורמטיבי מקסימלי, ומפצלים לפיה. הפיצול יכול להיות לשניים או יותר צמתים.
ממשיכים בתהליך עד שמגיעים לתנאי העצירה – כל הרשומות בצומת הנוכחי סווגו, או שהגענו לתנאי עצירה שהוגדר מראש כמו גודל העץ, ובמקרה זה הצומת תסווג לפי הערך של רוב הרשומות בו.

במימוש שלנו בחרנו max_depth=3 וזאת כי עץ עמוק יותר עשוי לעשות overfitting, במיוחד על datasets קטנים (שלנו מאוד קטן), כך שהגדרת מגבלה על העומק יכולה לעזור למנוע בעיה זו. בנוסף, עומק של 3 יכול לעתים קרובות ללכוד כמה מהדפוסים החשובים בנתונים ועדיין ניתנים לפירוש.

Random Forest:

האלגוריתם מתחיל בשורש העץ, עם סט הנתונים המלא.
בכל שלב עוברים על כל התכונות ולכל תכונה מחפשים סף פיצול אופטימלי שתביא לחלוקה לשני קבוצות, לאחר מכן האלגוריתם בוחר את התכונה שתביא לפיצול אופטימלי עם תתי קבוצות בעלות חוסר דיוק מזערי ביחס להסתברות לחיזוי משתנה היעד.
הפיצול ממשיך באופן רקורסיבי עד לקבלת צומת שכל הרשומות בו סווגו, או להגעה לתנאי עצירה מוגדר.
לאחר בנייה העץ מתבצעת בנייה של עצים גזומים, והאלגוריתם בוחר מבניהם את העץ האופטימלי ששומר על יחס בין רמת דיוק לבין זמן ריצה וגודל עץ, ובצורה זו מנסה להימנע מהתאמת יתר.
במימוש שלנו בחרנו שיהיה ביער עשרה עצים (n_estimators=10) וכן max_depth=3 מאותן סיבות בערך כמו בCART.

ג+ד. תוצאות הניתוחים + הערכת מידת דיוק של כל שיטה

תוצאות למודל CART עבור משימת החיזוי – לחזות האם לחולה יש או אין ckd על סמך תכונות מסוימות.
ה – confusion matrix מראה כמה true positive , false positive, true negatives ו – false negatives נחזו ע"י המודל. דוח הסיווג מראה ציון f1 עבור כל מחלקה, את סוג המחלקות (במקרה שלנו 0 או 1 כלומר notckd או ckd) , מספר הדגימות בכל מחלקה ואת הדיוק של המודל, הדיוק של המודל על ה test set היה 96% מה שאומר שהוא חזה את המחלקה של 96% מהדוגמאות. ציון קאפה המשוקלל – 91% מצביע על הסכמה מהותית בין המחלקות שנחזו לבין המחלקות האמיתיות, ה – AUC – 97% מייצג את השטח מתחת לעקומת ROC, מדד למידת ההבחנה של המודל בין שתי מחלקות, 50% לAUC אומר שהמודל לא יותר טוב מניחוש רנדומלי, בעוד ש 100% אומר שהוא מושלם, לכן המודל שלנו קרוב לשלמות מהבחינה הזו. סה"כ מודל CART תפקד בצורה מעולה ואפשר בהחלט להשתמש בו לחיזוי.

=== CART Tree ===

:Confusion Matrix
[2 26]]
[[51 1]

:Classification Report
precision recall f1-score support

28 0.95 0.93 0.96 0
52 0.97 0.98 0.96 1

accuracy 0.96 80
macro avg 0.96 0.95 0.96 80
weighted avg 0.96 0.96 0.96 80

Accuracy: 0.9625
Weighted Kappa: 0.9169
AUC: 0.9712

בדומה להסבר על CART, המודל Random Forest חזה אפילו טוב יותר והשיג דיוק של 98% על ה test set, ציון קאפה משוקלל של 97%, ו – AUC של 99%.

=== Random Forest ===

:Confusion Matrix
[0 28]]
[[51 1]

:Classification Report
precision recall f1-score support

28 0.98 1.00 0.97 0
52 0.99 0.98 1.00 1

accuracy 0.99 80
macro avg 0.98 0.99 0.99 80
weighted avg 0.99 0.99 0.99 80

Accuracy: 0.9875
Weighted Kappa: 0.9728
AUC: 0.9986

ה. ניתוח והסקת מסקנות

השוואה בין המודלים

בהתבסס על תוצאות החיזוי של שני המודלים CART ו- Random Forest, שניהם תפקדו היטב על ה – dataset ושניהם בעלי דיוק גבוה במיוחד.
עם זאת, נראה שלמודל Random Forest יש ביצועים טובים יותר מבחינת דיוק וקאפה משוקללת, עם דיוק מושלם ב – test set וציון קאפה משוקלל גבוה יותר בהשוואה לCART.
בנוסף למודל Random Forest יש ציון AUCגבוה יותר בהשוואה לCART, מה שמצביע על ביצועים טובים יותר מבחינת דיוק הסיווג הכולל.
לכן, בהתבסס על תוצאות אלה, Random forest עשוי להיות בחירה טובה יותר עבור הdataset שלנו.

מסקנות והצעות לשיפור והמשך עבודה

מסקנות:
שני המודלים CART ו Random Forest- השיגו שיעורי דיוק גבוהים, כאשר מודל ה- Random Forest השיג ביצועים מעט טובים יותר. למודל Random Forest היה קצב דיוק מושלם על ה – training set, אבל זה יכול להיות סימן להתאמת יתר. למודל CART היה קצב דיוק נמוך יותר על ה – training set, אך הביצועים היו טובים יותר על ה test set, מה שמרמז על כך שהוא נוטה פחות להתאמת יתר. לשני המודלים היו ערכי דיוק גבוהים, וציון f1 מה שמצביע על ביצועים טובים בשתי המחלקות. ערכי ה -קאפה המשוקללים וה- AUC עבור שני המודלים היו גבוהים אף הם, מה שמצביע על התאמה טובה בין הערכים החזויים לערכים בפועל.

אפשר להתנסות עם hyperparameters שונים עבור המודלים, כגון שינוי העומק המרבי או מספר העצים במודל Random forest, כדי לנסות ולשפר את הביצועים שלהם. אפשר גם לנסות להשתמש בסוגים אחרים של מודלים, כגון Support Vector Machines (SVMs) או Neural Networks, כדי לראות אם הם יכולים להשיג תוצאות טובות עוד יותר. אולי כדאי גם לאסוף נתונים נוספים במידת האפשר, כדי לשפר את הדיוק וההכללה של המודלים.