Machine Learning

Final Project

Yedidya Even-chen, Roni Har Tuv

תוכן עניינים

תיאור המאגר

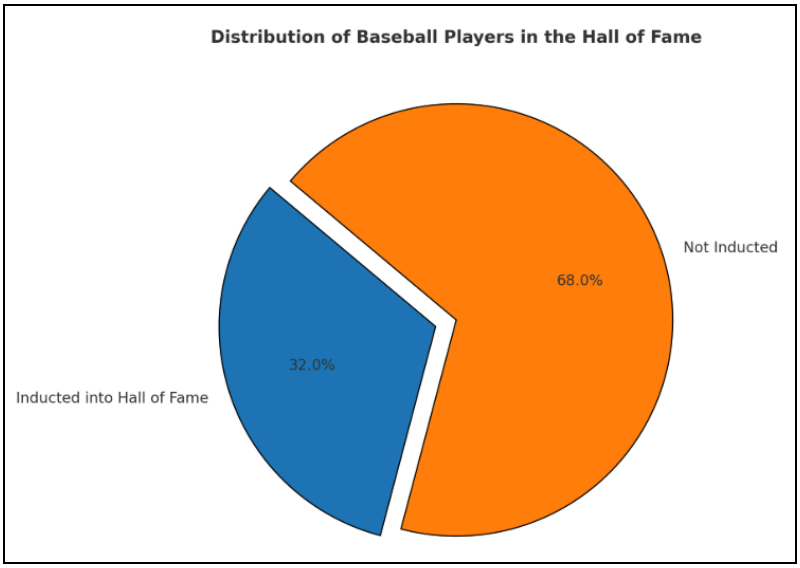
- קישור ישיר למאגר :
https://www.kaggle.com/datasets/krupadharamshi/500hits?select=500hits.csv

מאגר הנתונים **"מדדי שחקני בייסבול"** מכיל רשומות סטטיסטיות מקיפות עבור 500 שחקני בייסבול.
הוא כולל- שם השחקן , מספר השנים שבהן השחקן היה פעיל בליגת הבייסבול המקצועית, מספר משחקים ששיחק, מדדים מהותיים לביצועי השחקנים כגון- מספר חבטות (at-bats), ריצות שהובקעו, חבטות מוצלחות, מספר חבטות כפולות, חבטות משולשות, הום ראנס (home runs) ריצות שהוכנסו (RBI) וכו'.

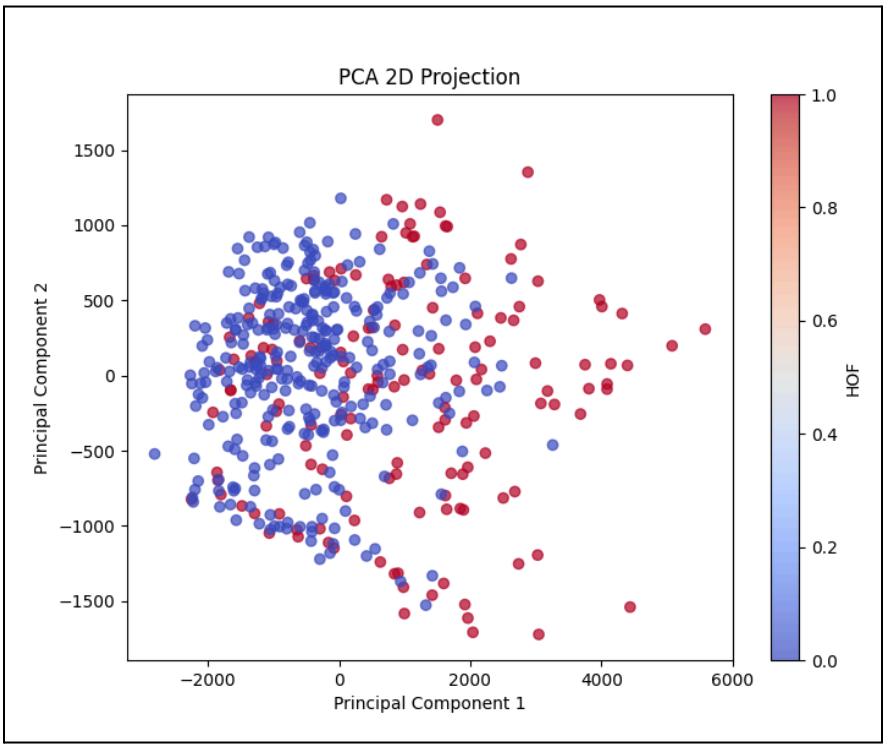
הנתונים משתרעים על פני מספר שנים וכוללים שחקנים ייחודיים, מה שמאפשר תובנות מעמיקות לגבי ביצועי שחקנים בודדים וכן השוואות בין תקופות וסגנונות משחק שונים. מאגר נתונים זה מהווה משאב בעל ערך רב לניתוח ומחקר בתחום הבייסבול.

להלן פירוט המידע הנתון עבור כל שחקן:

1. **PLAYER** - שם השחקן.
2. **YRS** - Years. מספר השנים שהשחקן שיחק.
3. **G** - Games. מספר המשחקים שהשחקן שיחק.
4. **AB** - At-bats. מספר הפעמים שהשחקן עלה לחבוט.
5. **R** - Runs. ריצות. מספר הפעמים שהשחקן הקיף את כל הבסיסים.
6. **H** - Hits. חבטות. מספר הפעמים שהשחקן הצליח לחבוט בכדור ולהגיע לבסיס הראשון.
7. **2B** - Doubles. פעמים שהשחקן הצליח לחבוט ולהגיע לבסיס השני.
8. **3B** - Triples. פעמים שהשחקן הצליח לחבוט ולהגיע לבסיס השלישי.
9. **HR** - Home Runs. פעמים שהשחקן הצליח לחבוט ולהגיע חזרה לבסיס הביתי.
10. **RBI** - Runs Batted In. מספר הריצות שקרו בזמן שהשחקן חבט.
11. **BB** - Base on Balls (Walks). פעמים שהשחקן קיבל בסיס ראשון "מתנה" בגלל 4 זריקות מחוץ לתחום.
12. **SO** - Strikeouts. פעמים שהשחקן לא הצליח לחבוט ב-3 זריקות.
13. **SB** - Stolen bases. בסיסים שהשחקן הצליח "לגנוב" (להתקדם "בלי רשות" בלי שנתפס).
14. **CS** - Caught stealing. פעמים שהשחקן נתפס תוך ניסיון לגנוב בסיס.
15. **BA** - Batting average. ממוצע חבטות. חבטות מוצלחות חלקי פעמים שעלה לחבוט.
16. **HOF** - Hall of Fame indicator (1 = in Hall of Fame, 0 = not). תיוג בינארי, האם השחקן בהיכל התהילה או לא.



התפלגות אחוזי הכניסה להיכל התהילה



פיזור הדגימות לאחר הורדת מימדים. ניתן לראות שיש ערבוב במידה גדולה יחסית.

הגדרת השאלה

בהינתן מאגר המידע, נרצה לחקור ולשאול את השאלות הבאות :

1. על סמך נתוני שחקן מסוים, האם הוא עתיד להיכנס להיכל התהילה של הבייסבול?
2. מהם המאפיינים המשפיעים ביותר על התשובה לשאלה 1?

הקוד

קישור לקוד בקולאב: `baseball.ipynb`

קישור לקוד בגיטהאב: Github

מבנה הקוד

לפני הרצת האלגוריתמים, יש שלב של preprocessing:

במאגר מדדי השחקנים, כל שחקן מיוצג על ידי סט של מאפיינים (features).
משתנה המטרה (Target Variable) יהיה משתנה בינארי המסמן האם השחקן נכנס להיכל התהילה.

נוריד מאפיינים שלא רלוונטים לתהליך הלמידה (שם השחקן).

נבצע נרמול של הנתונים לטווח [0, 1].

נבצע פיצול של המאגר לקבוצת אימון (training set) וקבוצת בדיקה (test set).

אחרי ה-preprocessing, הדאטא מסודרת בצורה שנוח לעבוד איתה.

נבדוק 4 מודלים מסווגים שונים.

אלגוריתמי סיווג שונים

אלגוריתם 1 - Logistic Regression

הסבר כללי על האלגוריתם

רגרסיה לוגיסטית היא מודל סטטיסטי המשמש לסיווג, כאשר המטרה היא לחזות את ההסתברות להשתייכות לאחת משתי קטגוריות. המודל מבוסס על פונקציית הסיגמוייד (Sigmoid Function), אשר ממירה ערכים רציפים להסתברויות בטווח [0,1].

המודל עובר אימון באמצעות Gradient Descent כדי למצוא את המקדמים האופטימליים שמקטינים את פונקציית עלות הנגזרת.

בשל אופייה הבינארי של השאלה שהגדרנו (שחקן נכנס להיכל התהילה או לא), רגרסיה לוגיסטית בינארית היא אלגוריתם מתאים, שכן היא מספקת הסתברות לכל שחקן בנוגע לסיווגו: "נכנס להיכל התהילה" (1) או "לא נכנס להיכל התהילה" (0).

יישום האלגוריתם ושלבי פעולה

תהליך העבודה כולל את השלבים הבאים:

1. יצירת המודל.
2. אימון המודל על ה-training set – חישוב המקדמים האופטימליים (maximum likelihood estimate), באמצעות Gradient descent.
3. בדיקת המודל על ה-test set כדי לקבל את מדדי הביצועים שלו.
4. **"בהינתן נתוני שחקן, האם הוא עתיד להיכנס להיכל התהילה?"** חיזוי הסתברויות – עבור שחקן חדש, חישוב ההסתברות לכניסתו להיכל התהילה. מעל 0.5 נכנס מתחת ל0.5 לא נכנס.

אלגוריתם 2 - KNN

הסבר כללי על האלגוריתם

K-Nearest Neighbors הוא אלגוריתם למידה מפוקחת (Supervised Learning) המשמש הן לסיווג (Classification) והן לרגרסיה (Regression).

האלגוריתם לא "לומד" במובן הרגיל של בחירת פרמטרים אופטימליים (כמו weight, bias) אלא הוא שומר את המידע הנתון ומשתמש בו כדי לייצר חיזוי עבור דגימה חדשה.

כאשר יש צורך לבצע חיזוי עבור דגימה חדשה, האלגוריתם מחשב את המרחק בינה לבין כל הדגימות בסט האימון. לאחר מכן, הוא בוחר את k השכנים הקרובים ביותר (לפי מרחק אוקלידי, מנהטן או מדדים אחרים). ההחלטה מתקבלת על פי רוב הקולות (Majority Voting) של השכנים הקרובים.

אם מדובר בסיווג בינארי (כמו במקרה שלנו), הדגימה תקבל את התווית של רוב השכנים.

יישום האלגוריתם ושלבי פעולה

תהליך העבודה כולל את השלבים הבאים:

1. יצירת המודל.
2. "אימון" המודל על ה-training set:
    a. בחירת k אופטימלי מתוך טווח מוגדר, על ידי השוואת הביצועים לכל k.
3. חישוב מרחקים – חישוב המרחק בין שחקן חדש לבין כל הדוגמאות בסט האימון.

בהינתן דגימה חדשה מתוך ה-test set:

4. קביעת סיווג – זיהוי k השכנים הקרובים ביותר ושיוך השחקן לקבוצה שבה הרוב הגדול.

אלגוריתם 3 - Random forest

הסבר כללי על האלגוריתם

Random Forest הוא אלגוריתם למידת מכונה המבוסס על ensemble learning. כלומר, הוא משלב מספר רב של עצי בחירה (Decision Trees) כדי לשפר את ביצועי הסיווג או הרגרסיה.

זהו אלגוריתם למידה מפוקחת (Supervised Learning) שבו כל עץ מקבל תת-מדגם של הנתונים (ע"י bootstrapping, כלומר החלוקה בין העצים מתבצעת עם חזרה, כלומר לא מובטח שכל דגימה תהיה בעץ אחד בדיוק). וכן תת-קבוצה של המשתנים (Feature Subsampling) כדי להפחית שונות ולמנוע התאמת-יתר.

כאשר נדרש חיזוי עבור דגימה חדשה, האלגוריתם בודק את הדגימה בכל העצים ומבצע רוב קולות (Majority Voting) עבור סיווג או ממוצע תחזיות עבור רגרסיה.

יישום האלגוריתם ושלבי פעולה

תהליך העבודה כולל את השלבים הבאים:

1. יצירת מספר רב של עצי החלטה – כל עץ מאומן על מדגם אקראי של הנתונים תוך בחירת רק חלק מהתכונות בכל צומת כדי לשפר גיוון.
2. כוונון פרמטרים – לדוגמה כיוון מספר העצים (n_estimators), העומק המרבי של העצים (max_depth), ומספר התכונות הנבחרות בכל צומת (max_features) כדי לשפר ביצועים.
3. חיזוי באמצעות רוב קולות – בסיווג, כל עץ מחזיר החלטה (האם השחקן ייכנס להיכל התהילה), וההכרעה הסופית מתקבלת על פי הרוב.
4. בדיקת הביצועים של המודל על ה-test set.
5. **"בהינתן נתוני שחקן, האם הוא עתיד להיכנס להיכל התהילה?"** לאחר אימון המודל, ניתן להזין נתוני שחקן חדש, וכל עץ ייתן תחזית משלו.
   באמצעות רוב קולות, המודל יקבע האם השחקן צפוי להיכנס להיכל התהילה,.

אלגוריתם 4 - SVM

הסבר כללי על האלגוריתם

Support Vector Machine הוא אלגוריתם למידת מכונה לסיווג ולרגרסיה, המבוסס על מציאת היפר-מישור (Hyperplane) אופטימלי שמפריד בין המחלקות בסט הנתונים.

המטרה של SVM היא למקסם את המרווח (Margin) בין הדגימות הקרובות ביותר להיפר-מישור, הנקראות וקטורי תמיכה (Support Vectors), וכך להבטיח הפרדה טובה בין הקבוצות.

יישום האלגוריתם ושלבי פעולה

תהליך העבודה כולל את השלבים הבאים:

1. יצירת המודל.
2. חישוב ההיפר-מישור האופטימלי עבור ה-training set – חיפוש המישור שמפריד בצורה הטובה ביותר בין הקבוצות, תוך מקסום המרווח.
   נשתמש ב-kernel כדי למפות את הנתונים למרחב ממימד גבוה יותר, על מנת למצוא הפרדה לינארית בין הדגימות.
3. חיזוי סיווג של דוגמאות חדשות מה-test set – הכנסת נתוני שחקן חדש ובדיקת מיקומו יחסית להיפר-מישור כדי לקבוע את תחזית הסיווג.
4. **"בהינתן נתוני שחקן, האם הוא עתיד להיכנס להיכל התהילה?"** לאחר אימון המודל, ניתן להזין נתוני שחקן חדש, והמודל יחשב את מיקומו ביחס להיפר-מישור.
   אם השחקן נמצא בצד החיובי של ההפרדה – הוא יסווג כמי שייכנס להיכל התהילה, ואם בצד השלילי – הוא לא ייכנס.

ניתוח והשוואת תוצאות

המדדים

1. Confusion matrix - היא כלי להערכת ביצועי מודל סיווג על ידי השוואת תחזיותיו לתוויות האמיתיות. היא מורכבת מארבע קטגוריות:

	Actual Negative השחקן לא בהיכל התהילה	Actual Positive השחקן כן בהיכל התהילה
Predicted Negative המודל חוזה שהשחקן לא יכנס להיכל התהילה	TN	FP
Predicted positive המודל חוזה שהשחקן כן ייכנס להיכל התהילה	FN	TP

2. **Precision** - מודד את שיעור הדגימות שסווגו נכון עם תיוג מסוים מתוך כלל הדגימות שהמודל חזה עבורן את התיוג הזה. ערך גבוה מצביע על כך שהמודל מפיק מעט חיוביים שגויים (False Positives) ומדויק בזיהוי המקרים החיוביים.

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

3. **Recall** - מודד את שיעור הדגימות שסווגו נכון עם תיוג מסוים מתוך כלל הדגימות שיש להן את התיוג הזה בפועל.
ערך גבוה מעיד על כך שהמודל מזהה היטב את התיוגים ומייצר מעט שליליים שגויים (False Negatives).

$$Recall = \frac{TP}{TP + FN}$$

4. **F1-Score** - הוא ממוצע הרמוני של Precision ו-Recall.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

5. **Accuracy** - משקף את אחוז התחזיות הנכונות מכלל הדגימות, ולכן הוא מדד כללי לאיכות המודל.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

כל אחד מהמדדים האלה מתקבל בנפרד עבור התיוג 0 והתיוג 1.

תוצאות האלגוריתמים

בפלט הקוד מופיעות תוצאות מפורטות על ביצוע האלגוריתם. כאן הבאנו את התוצאות הרלוונטיות לניתוח שלנו.

נתמקד בממוצעי macro כי אנחנו רוצים לייחס את אותה חשיבות למדדים של 1, למרות ה-imbalance.

ממוצע מאקרו מחשב את הממוצע של המדד משני התיוגים, כאשר לשניהם יש משקל שווה.

accuracy	F-1 score macro average	Recall macro average	Precision macro average	
0.81	0.76	0.74	0.79	LR
0.80	0.73	0.71	0.81	KNN
0.86	0.83	0.81	0.86	RF
0.82	0.79	0.79	0.79	SVM

מסקנות - תשובות לשאלות שלנו

1. האלגוריתם המוצלח ביותר עבור משימת הסיווג הוא Random Forest, ואחריו SVM.

לעומת זאת, KNN ו-Logistic Regression הציגו ביצועים פחות מרשימים, כנראה בשל רגישותם למבנה הנתונים ולמורכבות הדפוסים, אך אין לקבוע זאת באופן חד משמעי.

כמו שלמדנו בכיתה , אלגוריתם Random Forest נחשב מוצלח מאוד כאשר מדובר בקלסיפיקציה בינארית.

2. דירוג המאפיינים המשפיעים ביותר על החלטת הסיווג:

Feature		Importance
13	BA	0.153969
4	H	0.149490
3	R	0.126581
2	AB	0.086596
8	RBI	0.076090
1	G	0.073390
5	2B	0.065611
6	3B	0.054213
10	SO	0.047353
7	HR	0.045832
9	BB	0.041344
11	SB	0.027341
0	YRS	0.026763
12	CS	0.025428

ניתן לראות שהמאפיין המשמעותי ביותר הוא BA - batting average. בפשטות, זה מדד היחס של הצלחות מתוך נסיונות.

ה-BA של שחקן נחשב מדד אמין ליכולות שלו ולאיכות המשחק שלו, ולכם לא מפתיע שזה המדד המוביל בחשיבות.

אתגרים בדרך

- במאגר, כל עמודה מתייחסת למדד אחר, ולכן יש לנו 2 בעיות מרכזיות:
  - הנתונים לא מיוצגים באותה שיטה - רוב המדדים הם מספר טבעי כלשהו, ויש מדדים המיוצגים באחוזים (מספר בין 0 ל-1). קשה להשוות בין מדדים משני הסוגים האלו.
    לדוגמא - BA - מדד שכתוב באחוזים בין 0 ל-1 (ובפועל אין ערכים מעל 0.4). לעומת זאת HR מדד שכתוב במספר טבעי שאינו חסום.
  - גם אם המדדים מיוצגים באותה צורה (נגיד מספר טבעי), הטווחים משתנים בין העמודות.
    לדוגמה ב-CS יש ערכים בטווח 0-335, כאשר יש לא מעט שחקנים עבורם הערך הינו 0.
    לעומת זאת, ב-AB אין שחקן עם פחות מ-4000 והמספרים מגיעים עד 12,000.

  הפתרון שלנו לשתי הבעיות האלה הוא לבצע נרמול של הדאטא - כל המדדים יהיו בטווח בין 0 ל-1.
- מאגר קטן - בערך 500 שחקנים. בגלל שכמות השחקנים יחסית קטנה, כל תזוזה של שחקן תשפיע יחסית הרבה.
- מאגר לא מאוזן - בערך 32% בהיכל התהילה והשאר לא.
  זו אינה הטייה מאוד גדולה ומשמעותית, אך מודל שמנחש 0 תמיד צודק באחוזים סבירים. לכן, צריך לוודא שהמודל לא מפתח נטייה לא מאוזנת לכיוון ה-0.

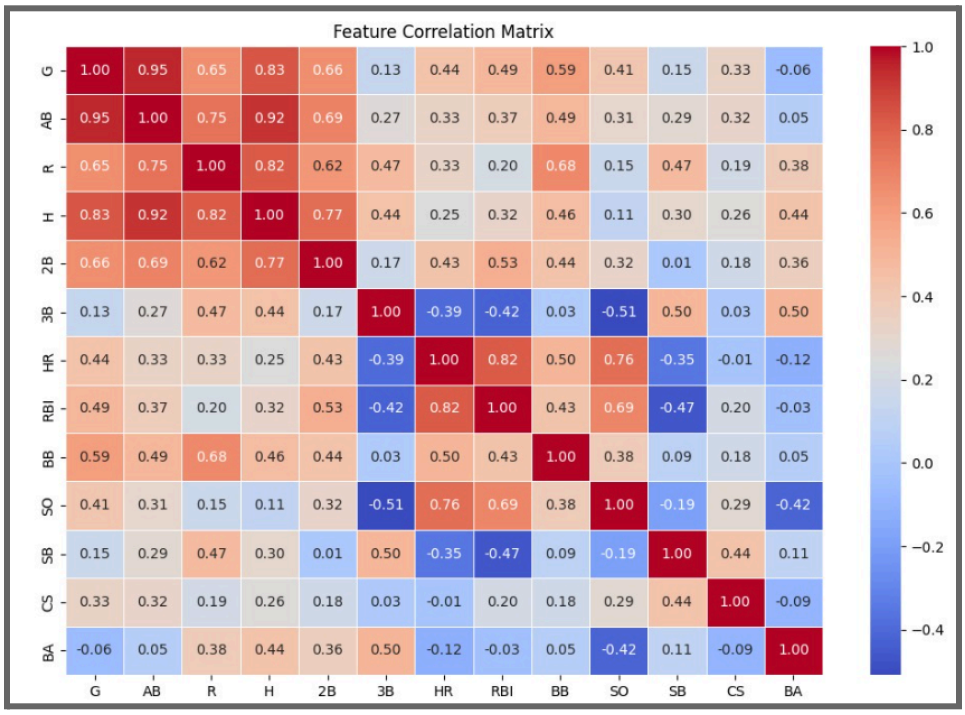
פתרון אפשרי לשתי הבעיות האלה הוא לבצע אוגמנטציה של המאגר - לייצר דגימות חדשות מתוך הדאטא הנתונה, כדי לשפר את האיזון בין התיוגים.

נפרט על הניסיונות לתקן את הבעיות האלה:

נסיונות שיפור

אפשרות אחת לשפר את ביצועי המודלים היא להוריד מאפיינים מיותרים כדי למנוע התאמת יתר ולשפר את איכות המודל באופן כללי.
במאגרים גדולים זה גם יכול לייעל את זמן הריצה, אך המאגר שלנו לא מספיק גדול כדי שזה יהיה משמעותי.

על מנת לנסות לשפר את ביצועי המודלים ניסינו לראות האם קיימים מאפיינים שנוכל לסנן ובכך להפחית את כמות העמודות במאגר, והאם פישוט זה יכול לשפר את הביצועים. קודם כל בדקנו קורלציה (התאמה) בין המאפיינים השונים:



בנוסף, נרצה לבדוק קורלציה של המאפיינים השונים עם השדה "נכנס להיכל התהילה", כי אם נרצה להסיר אחד מתוך 2 שדות שיש ביניהם התאמה גבוהה - נבחר להסיר את זה שפחות מתואם עם ההחלטה על היכל התהילה.

הסרת המאפיינים:

1. ישנה קורלציה גבוהה בין השדה AB לשדה G. השדה AB משפיע יותר על ההכרעה הסופית.
   לכן - נסיר את הפיצ'ר G

2. ישנה קורלציה גבוהה בין השדה H לשדה R. השדה H משפיע יותר על ההכרעה הסופית.
   לכן - נסיר את הפיצ'ר R

3. ישנה קורלציה גבוהה בין השדה HR לשדה RBI. השדה RBI משפיע יותר על ההכרעה הסופית.
   לכן - נסיר את הפיצ'ר HR

4. הפיצ'ר CS שואף ל 0 בהשפעה שלו על ההכרעה הסופית, ונמצא בפער מאוד גדול ביחס לפיצ'ר שמעליו (0.008672 מול 0.042447). לכן נסיר אותו.

5. איחוד של 2B 3B וHR- שדות אלו עם קורלציה בינונית, אך מבחינת משמעות המשחק ניתן לבצע חישוב שמאחד את הביצועים. נאחד אותם לשדה יחיד "B " שיחושב כך:

$$B := (H - 2B - 3B - HR) + (2 \cdot 2B) + (3 \cdot 3B) + (4 \cdot HR)$$

(החישוב נובע ישירות ממספר הנקודות הכולל שהשחקן משיג).

מסקנות מהרצת האלגוריתמים על המאגר לאחר צמצום המאפיינים

הנסיון הנ"ל לא הניב תוצאות טובות יותר באף אחד מהמודלים.

נסיק מכך שחוסר דיוק של המודל אינו נובע מהתאמת יתר, אלא משילוב של כמה דברים:

1. מאגר קטן יחסית.
2. חוסר איזון בין התיוגים.
3. דאטא שאינה ניתנת להפרדה בקלות.

hyperparameters

באופן כללי, כדי לשפר את כל אחד מהמודלים בדקנו hyperparameters שונים כדי למצוא את האופטימליים. הבחירות והתוצאות מפורטות בקוד.

את החיפוש עצמו ביצענו קודם ע"י חקירה ולימוד על הערכים המומלצים לכל מודל, וע"י ניסוי וטעייה.

אחרי שהבנו בערך מה הטווח הרלוטי, ביצענו grid search כדי למצוא את השילוב האופטימלי המדויק.

אוגמנטציה

כדי לפצות על חוסר האיזון בין התיוגים, ניסינו לבצע oversampling של המאגר כדי לייצר עוד דגימות בעלות תיוג 1.

ביצענו אוגמנטציה ע"י SMOTE (מייצר דגימות שדומות לדגימות המצויות).

זה אכן שיפר חלק מהביצועים - בפרט את הה-recall של 1 ואת ה-precision של 0, אבל פגעו ב- precision של 1 וב-recall של 0. ובאופן כללי, הביצועים היו גרועים יותר.

Under-sampling

אחת הבעיות של SMOTE היא שזה שכל הדגימות החדשות נגזרות מהמידע הקיים, ואפשר לטעון שזה מוריד את האמינות של המודל (כי הוא לא מתאמן רק על דגימות אמיתיות).

במקום לבצע over sampling, אפשר לבצע under sampling - לקחת רק חלק מהדגימות שיש להן את התיוג הנפוץ יותר, כדי שהמודל יתאמן על קבוצה מאוזנת.

ניסינו גם את זה. זה שיפר את ה-recall של 1, אבל פגע בכל שאר המדדים.

התוצאה הזו לא מפתיעה - באופן כללי מודל יהיה איכותי יותר ככל שהוא יתאמן על יותר דגימות.

נרמול

כפי שאמרנו, אחת הבעיות במאגר היא חוסר אחידות בין המאפיינים. ואכן בשלב הראשוני, הרצנו את המודלים על המאגר המקורי בלי נרמול, התוצאות היו גרועות למדי, והמודלים של SVM וריגרסיה לוגיסטית התקשו להגיע בכלל למודל מאומן.

יוצא הדופן הוא מודל random forest, שהגיע לאותן תוצאות בדיוק עם או בלי נרמול. בשני המקרים הוא היה המודל הכי איכותי.

מסקנות

התוצאות הכי טובות התקבלו על המאגר המקורי עם כל המאפיינים, אחרי נרמול, בלי over sampling או under sampling, עם מודל random forest.

באופן כללי - ככל שנתנו למודל יותר דאטא בלי שינויים (למעט נרמול) - הביצועים היו יותר טובים.