הטכניון – מכון טכנולוגי לישראל
הפקולטה למדעי המחשב

חיזוי תוצאות משחק טניס ודירוג שחקנים

פרוייקט בבינה מלאכותית - 236502

מרצה: פרופ' שאול מרקוביץ'

מגישות:

שחר כגן, 207992207, shacharkagan@campus.technion.ac.il

לינוי גנטי, 208536284, linoyganti@campus.technion.ac.il

תוכן עניינים

מבוא

"נפש בריאה בגוף בריא" – משפט זה, שנלקח מהיוון העתיקה, ממחיש את האידיאולוגיה של רבים מאיתנו. בתור סטודנטיות שאוהבות ספורט ועוסקות בענפי ספורט מגיל קטן ועד היום (לינוי בטניס ושחר בשחייה אומנותית), ידענו שאת הפרויקט שלנו נעשה בתחום שקשור לספורט.

בחרנו במשחק הטניס משום ששתינו מכירות את הכללים והחוקים, ועוקבות אחר התחרויות והשחקנים.

נתחיל עם קצת מבוא על טניס, מהם הכללים, מהו מהלך המשחק ואיך יודעים מי המנצח, נציין כמה מהתחרויות היוקרתיות ונסביר על שיטת הדירוג העולמית.

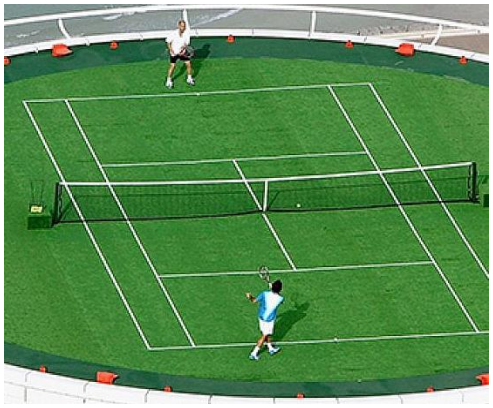
טניס הוא משחק כדור הנערך בין שני שחקנים (משחק יחידים) או שני זוגות שחקנים (משחק זוגות). מטרת המשחק היא לחבוט כדור גומי חלול באמצעות מחבט אל עבר צדו השני של המגרש.

הטניס (Tennis) הוא ענף ספורט פופולארי בכל רחבי העולם שזכה לכינוי "הספורט הלבן". ענף זה נולד בצרפת, ממשחק שנקרא בצרפתית ז'ה דה פום (Jeu de paume) אשר שוחק ע"י נזירים צרפתים שנהגו להשתעשע בחבטות בכדור עץ באמצעות ידם החשופה.

מקור השם טניס הוא במילה "טנה" (Tenez), שפירושה "קבל" בצרפתית עתיקה. הוא בא מהקריאה של השחקנים, שנהגו להגיש את מכת הפתיחה אל יריביהם.

מטרת משחק הטניס היא לחבוט בכדור באמצעות מחבט מעל הרשת אל צדו השני של המגרש, כך שהיריב לא יוכל לחבוט בכדור בחזרה אל שטח המגרש של יריבו. למשחק הטניס אין מגבלה של זמן, והוא נגמר כשאחד המתחרים זוכה בשתיים או בשלוש מערכות, בהתאם לסוג התחרות.

משחק הטניס מורכב ממשחקונים ומערכות.



משחקון מורכב מסדרת נקודות. בכל נקודות המשחקון חבטות ההגשה הן של אחד הצדדים, ובכל משחקון מתחלפת ההגשה לסירוגין. בכל משחקון יש ארבע נקודות, המקבלות ניקוד לפי הסדר: 15, 30, 40, 60. שחקן המגיע לניצחון הנקודה הרביעי מוכרז כמנצח במשחקון.

מערכה היא סדרת משחקונים. המתמודד הראשון שזוכה בשישה משחקונים, מוכרז כמנצח במערכה, אלא אם יריבו צבר חמישה ניצחונות במשחקונים, או אז יהיה עליו לזכות במשחקון השביעי בטרם יגיע יריבו לשישה ניצחונות.

המנצח במשחק נקבע לרוב לפי השיטה הנפוצה "הטוב מ-3 מערכות", הנהוגה בכל התחרויות לנשים ובחלק מהתחרויות לגברים. ישנה שיטה פחות נפוצה, "הטוב מ-5 מערכות", הנהוגה רק אצל הגברים בטורנירי: גראנד סלאם, בגביע דייוויס ובמשחקי גמר של טורנירים אחרים.

תחרויות הטניס מאורגנות בצורה של טורנירים המחולקים על פי מין ומשתתפים. ישנן תחרויות לגברים יחידים, נשים יחידות, ותחרויות לזוגות בהן יש שני שחקנים בכל צד של המגרש.

הגראנד סלאם הינה תחרות המכילה ארבעה טורנירים שונים הנחשבים לטורנירי הטניס היוקרתיים ביותר בעולם. כל אחד מהטורנירים נערך אחת לשנה. שמות הטורנירים הם: אליפות אוסטרליה הפתוחה, אליפות צרפת הפתוחה, אליפות ווימבלדון ואליפות ארה"ב הפתוחה.

התאחדות הטניסאים המקצועניים נקרא גם הסבב העולמי של הגברים בטניס, הוא סבב של טורנירים המנוהל על ידי התאחדות הטניסאים הידועה בראשי התיבות ATP. התאחדות הטניסאים המקצועניים מפרסמת מדי שבוע את הדירוג העולמי לטניסאים. הדירוג נקבע על פי ניקוד מוסכם מראש הניתן לכל שחקן על הישגיו בטורנירים בהם השתתף. העפלה לשלב מתקדם יותר בטורניר מדרגה גבוהה יותר מעניק לשחקן ניקוד גבוה יותר.

שיטת הדירוג החלה בשנת 1968, עם תחילת העידן הפתוח והפיכת הטניס למקצועני. החל משנת 1979 הפך הדירוג להיות על בסיס שבועי. נקודות הנצברות נספרות למשך שנה מרגע צבירתן, ובחישוב השבועי נלקחות בחשבון נקודות שהושגו ב-52 השבועות האחרונים.

דירוג נוסף בענף הטניס הינו ELO rating system. זוהי שיטת דירוג שמחשבת יכולת יחסית של מתמודדים במשחקי סכום אפס, בהם הפסד של מתחרה אחד הוא רווח של האחר. כאמור, היא נועדה במקורה לשפר את אופן הדירוג בשחמט אבל בהמשך נעשה בה שימוש גם בטניס והיתרון הבולט שלה הוא יכולתה לייצר השוואה בין ספורטאי עבר והווה באותו הענף, ואפילו בין ספורטאים ששיחקו באותו עידן והגיעו לפיק בתקופות שונות.

הצעת פרויקט

בפרוייקט זה נעסוק בכמה שאלות שמעניינות אותנו הקשורות לטניס:

1. בהינתן כל המידע שנצליח למצוא ברשת על משחק טניס בין שני שחקנים – מי ינצח
2. חיזוי תוצאות משחק בין שני שחקנים על סמך יכולות וביצועי השחקנים במהלך המשחק (כמות הגשות וכדומה – ללא ידיעת התוצאות בזמן אמת)
3. חיזוי תוצאות משחק בין שני שחקנים על סמך נתונים התחלתיים של המשחק (זהות השחקנים, סוג מגרש, תאריך...)
4. חיזוי דירוג ELO של שחקני טניס על סמך ביצועיהם.
   לדוגמה: האם שחקן X יהיה בין 20 המדורגים הראשונים בסוף 2020?

נפתור את השאלות הללו בעזרת בניית מודלי למידה כאשר הבעיה מורכבת ממספר פרמטרים משתנים.

נשתמש בשלושה מקורות מידע עיקריים:

- תוצאות כל המשחקים בתחרויות ה-ATP, WTA. בתוספת נתונים על המשחק ועל השחקנים. נתונים אלו נלקחו מפרוייקט הגיט https://github.com/JeffSackmann.
- תוצאות המשחקים בכל התחרויות בתוספת ההימורים המוקדמים לכל משחק. נתונים אלו נלקחו מ- http://www.tennis-data.co.uk/alldata.php.
- דירוגי ה-ELO של 150-250 השחקנים המובילים בכל שנה ורבעון. נלקחו מ- https://www.ultimatetennisstatistics.com/eloRatings.

אלגוריתמי הלמידה בפרויקט

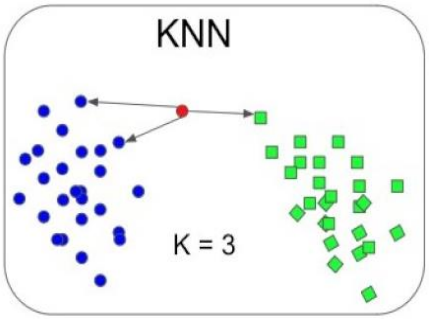
בלמידה ישנם שני סוגי בעיות: בעיות Classification ובעיות Regression.

בעיית Classification- בעיה בה המטרה היא לחזות תוצאה מתוך קבוצה סופית של ערכים הידועים מראש. במקרה שלנו 1 אם השחקן ניצח ו 0 אם הפסיד.

בעיית Regression- בעיה בה המטרה היא לחזות תוצאה מתוך טווח של ערכים רציפים. במקרה שלנו עבור שאלה 4 ניתן לכל שחקן ערך שמצביע על מספר הדירוג המשוערך, שהאלגוריתם צריך לנבות. פתרון הבעיה הוא בעזרת פונקציה $f(x)$ שתיבנה ע"י מודל הרגרסיה שנבחר. הפונקציה מהצורה:

$$y = f(x) + \epsilon \quad f \in F, \epsilon \sim D_{ERR}$$

KNN- K-Nearest Neighbors



איור 1: אלגוריתם KNN

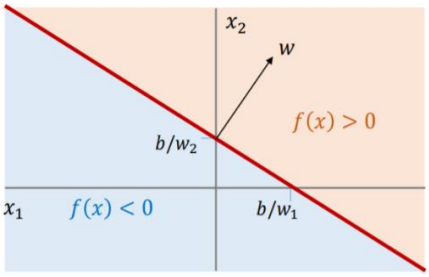
זהו אלגוריתם מבין הבסיסיים ביותר בתחום למידת המכונה. הוא מתבסס על ההנחה "אם שני אובייקטים דומים, גם הסיווגים שלהם יהיו דומים". כאשר דמיון בין אובייקטים יימדד לפי מרחק במרחב התכונות.

בשלב האימון רק נאחסן את הדוגמאות ואת הסיווג שלהן. בשלב הסיווג, נמדוד את הדמיון בין האובייקט הבלתי מסווג לכל דוגמאות האימון ששמרנו. הסיווג יקבע לפי החלטת הרוב של K הדוגמאות הכי קרובות (השכנים).

זמן אימון-אפס, זמן סיווג-כגודל קבוצת האימון (חישוב מרחק מכל דוגמא). האימון מהיר, אבל הסיווג יקר.

<u>במסווג זה נבחן את הפרמטרים:</u>
n_neighbors – מספר השכנים לפיו נסווג.

SVM-Support Vector Machine

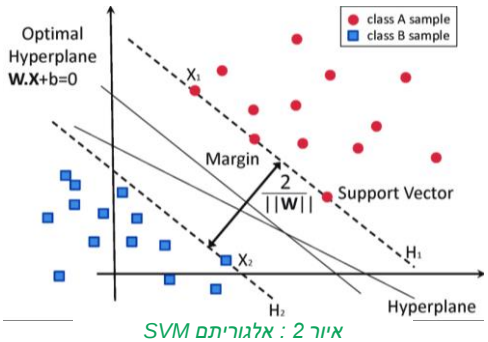


איור 3: אלגוריתם SVM

בשלב האימון נתאים מסווג אשר מפריד באמצעות ווקטור לינארי (בדר"כ ליניארי אלא אם נעשה שימוש ב-kernel) בין דוגמאות אימון חיוביות לשליליות. המסווג שנוצר, יוצר מרווח (margin) גדול ככל האפשר בינו לבין הדוגמאות הקרובות לו ביותר בשתי הקטגוריות. בשלב הסיווג, כאשר מוצגת דוגמה חדשה, האלגוריתם יזהה באיזה צד של המפריד הדוגמה נמצאת ויסווג בהתאם.

מציאת הווקטור המפריד (יקרא w) מתבצעת ע"י פתירת משוואת אופטימיזציה שמקטינה את כמות/גודל הטעויות (לפי פונ' loss) ומגדילה ככל הניתן את ה-margin:

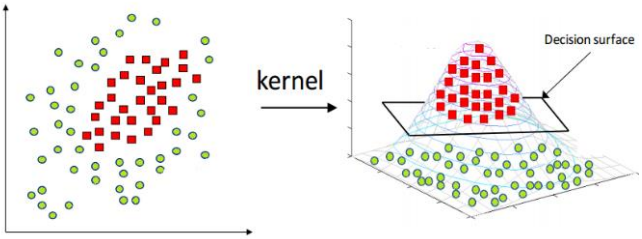
$$\min_{w \in \mathbb{R}^d, \xi \in \mathbb{R}^m} ||w||_2^2 + C \sum_{i \in [m]} \xi_i \quad s.t\ y_i \cdot w^T x_i \geq 1 - \xi_i, \forall i \in [m]: \xi_i \geq 0$$



איור 2 : אלגוריתם SVM

שימוש בקרנל:

הווקטור שמקבלים מפתירת המשוואה הוא בעצם צירוף לינארי של מכפלות פנימיות של הדוגמאות הקיימות $(x \cdot x')$, ולכן ניתן להחליף את הדוגמאות בטרנספורמציה כלשהי שלהם $K(x,x')$ (לדוגמה, העלאת כל הערכים בריבוע) ועדיין זמן החישוב יישאר דומה. שימוש בקרנל (הכלת טרנספורמציה) יכול לתת ביטוי חדש להפרדה כזו.



איור 4 : אלגוריתם SVM עם Kernel

<u>במסווג זה נבחן את הפרמטרים:</u>

C – מקדם רגולריזציה, הינו מאזן את החשיבות בין הפרדה גבוה של הדאטה לבין margin גדול. הערכים שנבדוק ינועו בין 10^{-5} לבין 10^{5}.

הקצה התחתון בטווח נותן דגש להפרדה גבוה, כך שהדוגמאות יהיו בשני צדדים שונים של הווקטור, גם על חשבון margin קטן. למסווגים מסוג זה קוראים גם hard-svm שכן הם לא מאפשרים טעויות כלל. לרוב ינטו ל-overfitting שכן מתאימים את עצמם מאוד לסט האימון.

הקצה העליון בטווח נותן דגש לmargin גדול ככל הניתן, כך שטעויות בסט האימון מתאפשרות. הגדלת הmargin נותנת גמישות לאלגוריתם ומונעת התאמה מדויקת של המפריד לסט האימון, אך מקדם רגולריזציה גדול מידיי יגרור ל $w = 0$ וזה בעצם underfitting.

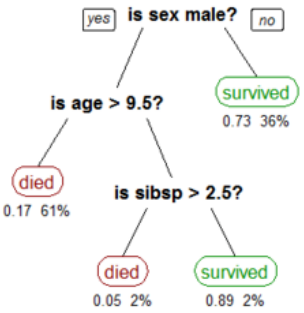
Kernel poly – קרנל המעלה את המכפלה הסקלרית של כל זוג דוגמאות במעלה כלשהי: $K(x,x') = (x^T x')^d$. פרמטר ייחודי שבחנו לקרנל הינו Degree – המעלה שאיתה מעלים את המכפלות הסקלריות בין הדוגמאות.

Kernel rbf – קרנל המחשב לכל זוג דוגמאות את הפונקציה: $K(x,x') = \exp\left(-\gamma\left||x - x'|\right|^2\right)$.
פרמטר ייחודי שבחנו לקרנל הינו Gamma – אותו הפרמטר שניתן לראות בפונקציית הקרנל.

Decision Tree

עץ החלטה הוא אלגוריתם המבוסס על בניית עץ בו כל צומת הוא שאלה המובילה לצומת אחרת (השאלה הבאה) או לעלה המספק תשובה סופית (סיווג מסוים).

בניית העץ תתבצע ע"י בחינת התכונה שתגרום לinformation gain יותר גדול ע"י חישוב ה- entropy, ופיצול הצומת לפיה.



איור 5 : אלגוריתם DECISION TREE

<u>במסווג זה בחנו את הפרמטרים:</u>

Max_depth – עומק העץ. נחסום אותו על מנת להימנע מהתאמת יתר(overfitting) לסט האימון.

Random Forest

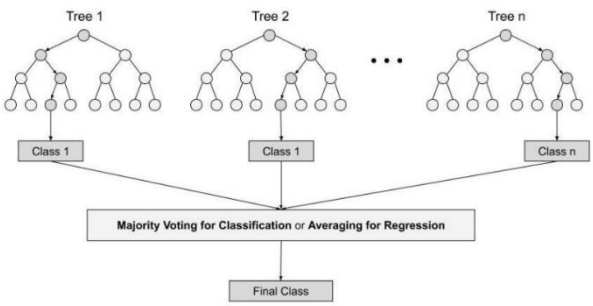
אלגוריתם אשר משתמש במספר עצי החלטה, על מנת לשפר את דירוג הסיווג.

כיוון שעץ החלטה בודד בעל שונות (variance) גדולה, שכן העץ מסתעף ככל שיש יותר תכונות ויותר דוגמאות, הוא מועד יותר לoverfitting. בעת שימוש במספר של מסווגים חלשים יותר

נקטין את השונות ובכך נמנע התאמת יתר של האלגוריתם לסט האימון.

האלגוריתם בונה מס' עצים, בבנייה שהיא זהה לעץ החלטה בודד, אבל הדוגמאות והתכונות שנלמדות, שונות בין עץ לעץ. כל עץ מקבל דוגמאות המוגרלות עם חזרות מכלל דוגמאות האימון וסט רנדומלי של תכונות.

הסיווג מתבצע לפי החלטת רוב העצים ביער.



איור 6: אלגוריתם Random Forest

<u>במסווג זה נבחן את הפרמטרים:</u>

Criterion – הפונקציה שמצביעה על טיב החלוקה של הדוגמאות בצומת. קיימות שתי סוגי פונקציות: gini, entropy.

n_estimators – מספר העצים שישתתפו בבניית היער.

max_features – מספר התכונות שבוחנים כאשר רוצים לבצע חלוקה. אנו נבחן את sqrt, log2.

max_depth – העומק המקסימלי של כל עץ ביער.

min_samples_split – מספר הדוגמאות המינימלי הדרוש בצומת על מנת לחלק את הצומת.

min_samples_leaf – מס' הדוגמאות המינימלי הנדרש על מנת להפוך צומת לעלה.

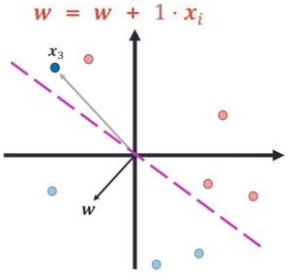
Perceptron

```
w = 0_d

while didn't separate trainset
    for i=1 to m
        ŷ_i = sign(w^T x_i)

        if y_i != ŷ_i
            w = w + η y_i x_i
```

$w = w + 1 \cdot x_i$



איור 7: אלגוריתם Perceptron

זהו אלגוריתם שמוצא מסווג תוך כדי ריצה על הדוגמאות מקבוצת האימון עד למציאת מפריד לינארי w המפריד בין הדוגמאות. אם הוא לא מוצא מפריד כזה, כלומר אין מפריד לינארי לדאטה, האלגוריתם ימשיך לרוץ ללא הפסקה "סתם" ולכן נהוג להגביל אותו בזמן.

נציין כי למרות שגם הוא וגם SVM מחפשים מפריד לינארי הם לא בהכרח נותנים את אותו מפריד.

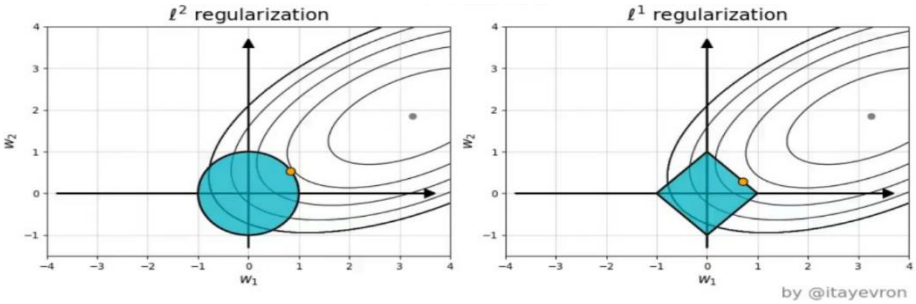
<u>במסווג זה נבחן את הפרמטרים:</u>

eta0 - זה η שבנוסחה, כלומר אם ה-w אינו נכון אז נשנה אותו ב-η כפול השגיאה, כלומר זה עוצמת ההתקדמות למפריד של הדאטה.

Penalty – הפונקציה לפיה תחושב הרגולריזציה ותמנע מווקטור המשקלים שהאלגוריתם מחפש להגיע לערכים גבוהים ב-w שיגרום למצב של לoverfitting.

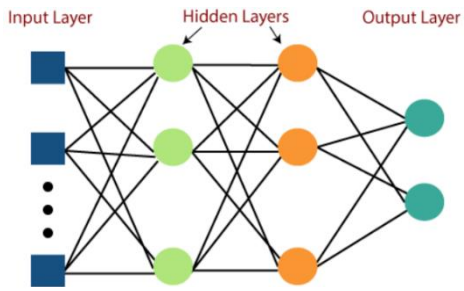
נבחן 4 פונקציות: L2 (נורמת l2 – מרחק אוקלידי), L1 (נורמת l1), elasticnet (שילוב של שתי הנורמות), ו-None (ללא שימוש בפונקציות רגולריזציה כלל).

max_iter- מקסימום האיטרציות שיש לבצע על מנת ללמוד. נקבע ערך זה להיות 1500 באופן קבוע.



איור 8: הדגמה להבדל בין L1 ל L2

MLP – Multi Layer Perceptron



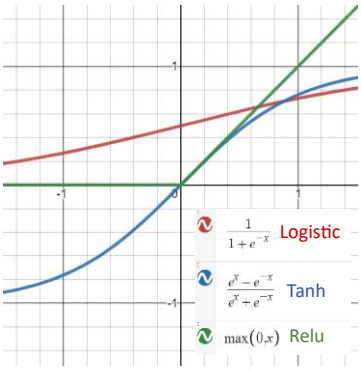
איור 9: MLP- Multy Layer Perceptron

אלגוריתם זה משתמש בפרספטרונים במספר שכבות, כשבין כל שכבה יש לנו פונקציית אקטיבציה שגורמת למפריד להיות לא לינארי. זה גורם לדיוק יותר גבוה כיוון שלא כל דטאה הינו פריד ליניארי.

<u>במסווג זה נבחן את הפרמטרים:</u>

hidden_layer_sizes – מספר השכבות הנסתרות וכמות הנוירונים בכל שכבה.

Activation – פונקציית האקטיבציה. נבחן 3 פונקציות: Logistic, Tanh ו- Relu.



איור 10: פונקציות האקטיבציה

Solver – השיטה שאיתה פותרים את בעיית האופטימיזציה של הפרספטרון. נבחן את שיטת ה-sgd שתוסבר למטה, ואת שיטת adam, אלגוריתם המתבסס על sgd אך בעל שני מקדמי למידה ולא אחד ולכן נגזרת התקדמות שונה.

Alpha – מקדם הלמידה. הערכים שנבדוק ינועו בטווח בין 10^{-5} לבין 10^{2}.

max_iter - מקסימום האיטרציות שיש לבצע בכל Perceptron יחיד על מנת ללמוד. הערכים שנבדוק ינועו בין 200 לבין 1000.

AdaBoost

אלגוריתם בו לוקחים היפותזות חלשות ומריצים אותן אחת אחרי השנייה כאשר כל אחת משפיעה על הבאה אחריה. לדוגמאות בהן אנו שוגים במסווג מסוים נותנים חשיבות גדולה יותר כך שהמסווג הבא יבין שהוא צריך לא לטעות בדוגמאות אלו. תוצאת הסיווג נקבעת לפי סכום ההיפותזות כאשר הינן ממושקלות.

אנו נבחן את האלגוריתם עם מסווגים חלשים שהם גדמי עץ החלטה, כלומר עצי החלטה עם עומק 1.

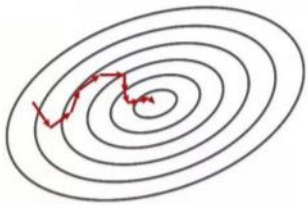
<u>במסווג זה נבחן את הפרמטר:</u>

n_estimators – מס' המסווגים החלשים שנבנה, ומס' האיטרציות הכולל שנעשה באלגוריתם.

Boosting

שיטה אשר לוקחת מסווגים חלשים ומייצרת מהם מסווג חזק, קיימות מס' שיטות ללמידה זו (random forest הוא אחד מהם)

GD-Gradient Descent


איור 11 : התקדמות Gradient Descent

אלגוריתם אופטימיזציה איטרטיבי. זוהי שיטה אשר נועדה למזער פונקציה עם מספר משתנים ולכן משתמשים בה כדי למזער את פונקציית ההפסד. תחילה האלגוריתם מריץ את המודל עם משקולות ראשוניות, ואז מבקש למזער את פונקציית ההפסד על ידי עדכון המשקולות על פני מספר איטרציות.

SGD-Stochastic Gradient Descent

אלגוריתם הפועל באותו אופן כמו GD, אך במקום לבצעו על כל קבוצת הדוגמאות נבחר תת קבוצה של דוגמאות ועליהם נריץ GD על מנת למזער את זמן הריצה.

SGD- Regression

מודל המחפש פונקציה ע"י ביצוע SGD על המרחק בין הדוגמאות לתוצאה של הפונקציה.

<u>במודל זה נבחן את הפרמטרים:</u>

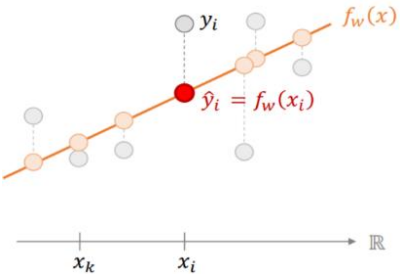
l1_ratio - מה היחס בין L1 ל L2 מבחינת רגולריזציה. 0 פרושו רגולריזציה רק עם L2 ו1 פרושו רגולריזציה רק עם L1 .

Alpha – מקדם הלמידה. הערכים שנבדוק ינועו בין 10^{-5} לבין 10^{5}.

Ridge regression (L2 regularization)

מודל המחפש פונקציה $f_w(x)$ שתנסה להגיע למרחק המינימלי מהתיוג האמיתי של הדוגמאות הנלמדות. כלומר לפתור את המשוואה הנ"ל:

$$\bar{w} = arg\min_{w} ||Xw - y||_2^2 + \lambda ||w||_2^2$$



איור 12:מציאת פונקציית מטרה במודל Ridge

$||Xw - y||_2^2$ ← אחראי להקטין את השגיאה של הפונקציה שתבחר.

$\lambda ||w||_2^2$ ← רגולריזציית L2 מגבילה את w על מנת למנוע overfitting.

<u>במודל זה נבחן את הפרמטרים:</u>

λ – מקדם רגולריזציה, הינו מאזן את החשיבות בין הפרדה גבוה של הדאטה לבין margin גדול. הערכים שנבדוק ינועו בין 10^{-5} לבין 10^{8}.

Kernel poly – קרנל המעלה את המכפלה הסקלרית של כל זוג דוגמאות במעלה כלשהי: $K(x, x') = (x^T x')^d$. פרמטר ייחודי שבחנו לקרנל הינו Degree – המעלה שאיתה מעלים את המכפלות הסקלריות בין הדוגמאות.

Kernel rbf – קרנל המחשב את הפונקציה: $K(x, x') = \exp\left(-\gamma ||x - x'||^2\right)$. פרמטר ייחודי שבחנו לקרנל זהGamma: – אותו הפרמטר שניתן לראות בפונקציית הקרנל.

תיאור המערכת שאלות 1-3

נבנה ממשק שיקבל מהמשתמש פרטים אודות משחקי טניס ויחזיר מי השחקן המנצח.

דוגמת הרצה:

```
Command Prompt
Microsoft Windows [Version 10.0.19043.1288]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>cd repos\ai_project

C:\Users\user\repos\ai_project>python our_tennis_predictor.py
Welcome to tennis match predictor by Linoy and Shachar!
Choose question to predict (1-3), for explenations of the questions, press h
3
You chose the third question! Let's start
Enter name of the first player: <First_name> <Last_name> (press Enter for autocompleter)
roger fed
Enter name of the second player: <First_name> <Last_name> (press Enter for autocompleter)
rafael nadal
Enter the match date in YYYY-MM-DD format: 2013-02-27
The match is best of 3 or 5?: 3
Enter number of players in the draw: 8
Enter the surface type (1: Carpet, 2: Clay, 3: Grass, 4: Hard): 2
Enter the tourney level:
1: Grand Slams,
2: Masters 1000s,
3: Challengers,
4: Satellites/ITFs,
5: Davis Cup,
6: other tour-level events,
7: Tour finals and other season-ending events
3
Enter the round of the match:
1: Final round,
2: Semi-Final round,
3: Qualification round,
4: Round of 16,
5: Round of 32,
6: Round of 64,
7: Round of 128
8: Round robin ("RR")3
Our model predicts the winner is ←[1mRafael Nadal←[0m!

C:\Users\user\repos\ai_project>
```

על מנת לבנות את המודל החוזה שלנו, נבצע מס' שלבים שיפורטו לפרטים בפרקים הבאים:

- בניית טבלה גדולה מכלל מקורות הדאטה שאספנו
- חלוקת הדאטה לקבוצות אימון ומבחן
- ביצוע ניתוח והכנת הדאטה לאימון
- ביצוע ניסויים לכוונון פרמטרים למודלי למידה
- החלת עיבוד הנתונים על סט המבחן
- בדיקת דיוק המודלים על קבוצת המבחן ובחירת מודל סופי
- החלת המודל בממשק שלנו

הכנת ה-DATA שאלות 1-3

ניצור טבלה אחת גדולה מכל המידע שאספנו שתכלול מס' תכונות אשר מחולקות לחמש קטגוריות.

א. <u>תכונות יבשות אודות המשחק</u>

1. tourney_name - שם התחרות
2. tourney_id - מספר ייחודי לתחרות
3. match_num - מספר ייחודי למשחק בתוך התחרות
4. Surface - סוג המגרש עליו מתרחש המשחק. ישנם 4 סוגי מגרשים עליהם מתחרים בטניס: חימר, דשא, משטח קשה (בדר"כ אספלט) ומשטח סינטטי (Carpet) שהיה בשימוש עד 2009.
5. Court - האם המשחק היה במגרש פתוח או מקורה
6. draw_size - מספר השחקנים בהגרלה, שלרוב מעוגל מעלה (חזקה) הקרובה ביותר מלמעלה של 2 (לדוגמה, טורניר עם 28 שחקנים עשוי להופיע כ- 32, כלומר 2^5)
7. tourney_date- תאריך קיום התחרות
8. tourney_level - סוג הטורניר. עבור גברים:
   'G' = Grand Slams
   'M' = Masters 1000s
   'A' = other tour-level events
   'C' = Challengers
   'S' = Satellites/ITFs
   'F' = Tour finals and other season-ending events
   'D' = Davis Cup
9. best_of-סוג המשחק, הטוב מ- 3 או 5 (מס' המערכות במשחק).
10. Round - שלב בתחרות, משלב המוקדמות דרך שלב הבתים ועד הגמר (RR, BR, R128, R64, R32, R16, QF, SF, F)

ב. <u>תכונות אודות השחקנים: i=1,2</u>

1. pi_id - ת"ז של השחקן ה-i
2. pi_name - שם השחקן ה-i
3. pi_seed - מספר שניתן לשחקן שמצביע על דירוג ראשוני לצורכי הגרלה (על מנת שרמת השחקנים בכל בית תהיה מגוונת)
4. pi_entry – מערכת דירוגים שקובעת האם שחקן יעלה ישירות ללא השתתפות במוקדמות
5. pi_hand - היד החזקה של השחקן ה-i (R= right, L= left, U= unknow)
6. pi_ht - גובה של השחקן ה-i בסנטימטרים
7. pi_age - גיל השחקן ה-i
8. pi_ioc - מדינה אותה מייצג השחקן ה-i
9. pi_atp_rank - הדירוג של השחקן ה-i לפי ATP ביום המשחק
10. pi_rank_points - מס' הנקודות של השחקן ה-i לפי חישוב ATP
11. pi_elo_rank - הדירוג של השחקן ה-i לפי ELO ביום המשחק
12. pi_elo_points - מס' הנקודות של השחקן ה-i לפי חישוב ELO
13. pi_elo_bestRank - הדירוג הכי גבוה במדד הELO של השחקן ה-i לאורך כל הקריירה שלו
14. pi_elo_bestPoints - מס' הנקודות הכי גבוה במדד הELO של השחקן ה-i לאורך כל הקריירה שלו

ג. <u>תכונות אודות תוצאת המשחק וניתוח המשחק והשחקנים: i=1,2</u>

1. Minutes- משך המשחק בדקות
2. pi_ace - מס' ה- aces שעשה השחקן ה-i במשחק

אייס (ace) – חבטת הגשה ('סרב') הנוחתת בשטח השחקן היריב, שבה היריב אינו מצליח לגעת בכדור. מזכה את השחקן המגיש בנקודה. לרוב חזקה, ונוטה לנחות באחת הפינות של אזור ה'סרוויס בוקס'.

3. pi_df- מס' ה-doubles faults שעשה המנצח במשחק

דאבל פולט (double fault) – החטאה כפולה ברציפות מצד שחקן הטניס המגיש.
דוגמה: הכדור נעצר ברשת, או שהחבטה חזקה מדי ויוצאת מן השטח המותר. במצב כזה, היריב מקבל את הנקודה.

4. pi_svpt - מס' ההגשות שעשה השחקן ה-i
5. pi_1stIn - מס' ההגשות הראשונות שנכנסו במהלך המשחקים לשחקן ה-i
6. pi_1stWon - מס' הניצחונות בהגשה הראשונה של השחקן ה-i
7. pi_2ndWon - מס' הניצחונות בהגשה השנייה של השחקן ה-i
8. pi_SvGms - מס' הגשות במשחק עבור השחקן ה-i
9. pi_bpSaved - מס' נקודות השבירה שהשחקן ה-i הצליח להשיג במשחק

נקודת שבירה (break point) - מצב בו חסרה לשחקן נקודה אחת על מנת לנצח משחקון, שבו השחקן היריב הוא המגיש ולכן נהנה מיתרון מובנה.

10. pi_bpFaced – מספר הפעמים שהשחקן ה-i פספס מצב של נקודת שבירה לא לטובתו.

ד. <u>תכונות אודות הימורים על המשחק:</u>

אספנו מידע על הסיכוי של player1, player2 לזכות במשחק לפני אתרי הימורים.
להלן הסבר על ראשי התיבות של ההימורים ולאיזה אתר הם שייכים:

1. CB - Centrebet odds of match
2. GB - Gamebookers odds of match
3. EX - Expekt odds of match
4. B&W - Bet&Win odds of match
5. LB - Ladbrokes odds of match
6. IW - Interwetten odds of match
7. SB - Sportingbet odds of match
8. PS - Pinnacles Sports odds of match
9. SJ - Stan James odds of match
10. UB - Unibet odds of match
11. B365 - Bet365 odds of match

ה. <u>תכונות אודות תוצאות זמן אמת:</u>

1. תוצאות המשחק בפורמטר: X-Y(Z) עבור כל משחקון

חילקנו אותן לתוצאות של כל שחקן בכל מערכה $j = 1,2,3,4,5$. (כלומר, תכונה אחת הפכה ל-15 תכונות)

p{i}_set{j}_score – תוצאת מערכה j עבור שחקן i.

set{j}_breakpoint_score - תוצאת נקודות ה-breakpoint במערכה j .

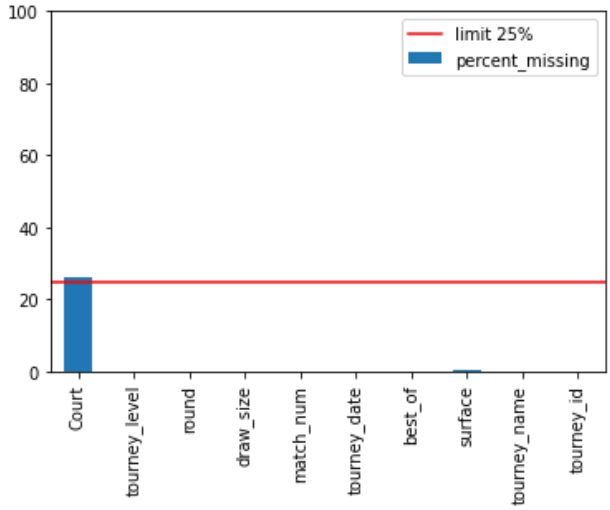
ניתוח הדאטה

לפני שננתח את המידע שאספנו ונבצע עליו טרנספורמציות מסוימות, נחלק את המידע לקבוצת אימון ולקבוצת מבחן כך שקבוצת האימון הינה 80% מכל הדוגמאות וקבוצת המבחן הינה השאר. קבוצת המבחן תשמש אותנו לבחינת דיוק האלגוריתמים שבחרנו לבחון. במידה ולא היינו מחלקות בשלב זה אלא בשלב מאוחר יותר הדיוק על המבחן יושפע גם מההכנה של הדאטה שנבצע בשלב הבא.

קיבלו סט אימון עם 47500 דוגמאות וסט מבחן עם 11900 דוגמאות.

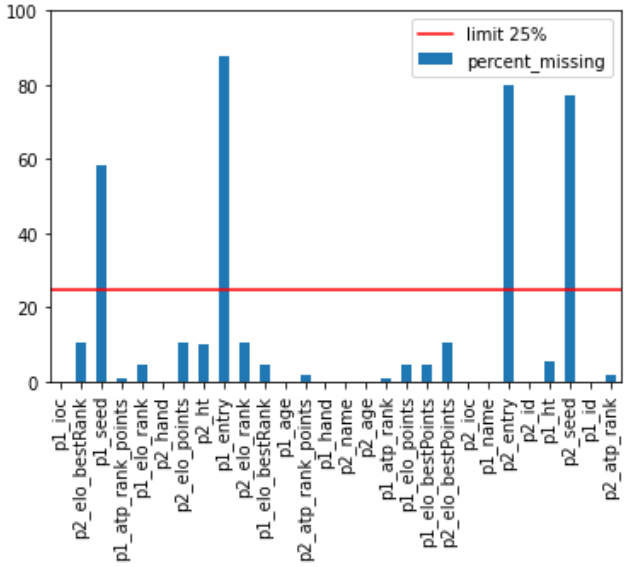
נתחיל בניתוח הטבלה שלנו, ניתן לראות כי ישנן הרבה תכונות.

נבדוק האם קיימות תכונות שהרבה ערכים בהן מכילים מידע חסר. נקבע כי עבור כל תכונה בעלת מעל 25% ערכים חסרים נמחק אותה. נציג זאת במספר גרפים לפי הקטגוריות. הגרף מציג (בעמודה כחולה) את אחוז הערכים החסרים עבור כל תכונה.



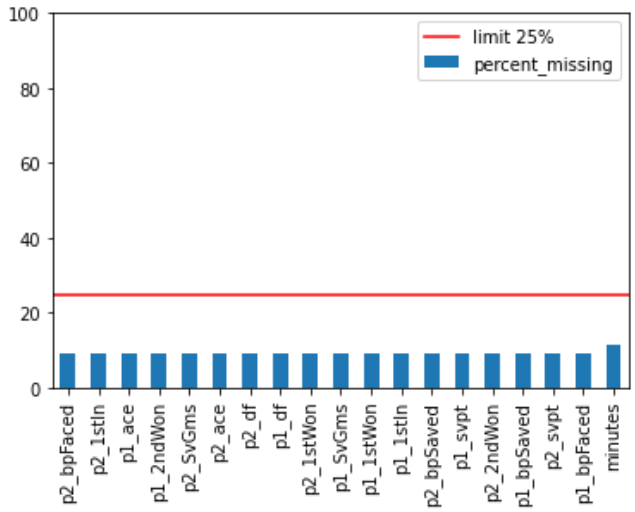
גרף 1: אחוז ערכים חסרים עבור קטגוריה א

עבור תכונה Court ישנם 26.3% ערכים חסרים ולכן נמחק אותה מהטבלה.



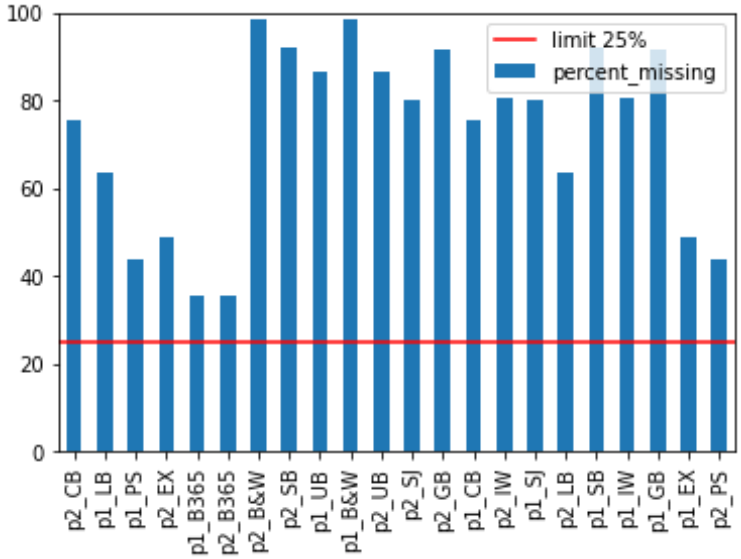
גרף 2: אחוז ערכים חסרים עבור קטגוריה ב

עבור התכונות: p1_seed(58.1%), p1_entry(87.8%) p2_seed(77.1%), p2_entry(80.1%) ישנם מעל 25% ערכים חסרים ולכן נמחק אותם מהטבלה.



גרף 3: אחוז ערכים חסרים עבור קטגוריה ג

בגרף זה אין תכונה בעלת מעל 25% ערכים חסרים ולכן לא נוריד שום תכונה מהטבלה עבור קטגוריה זו.



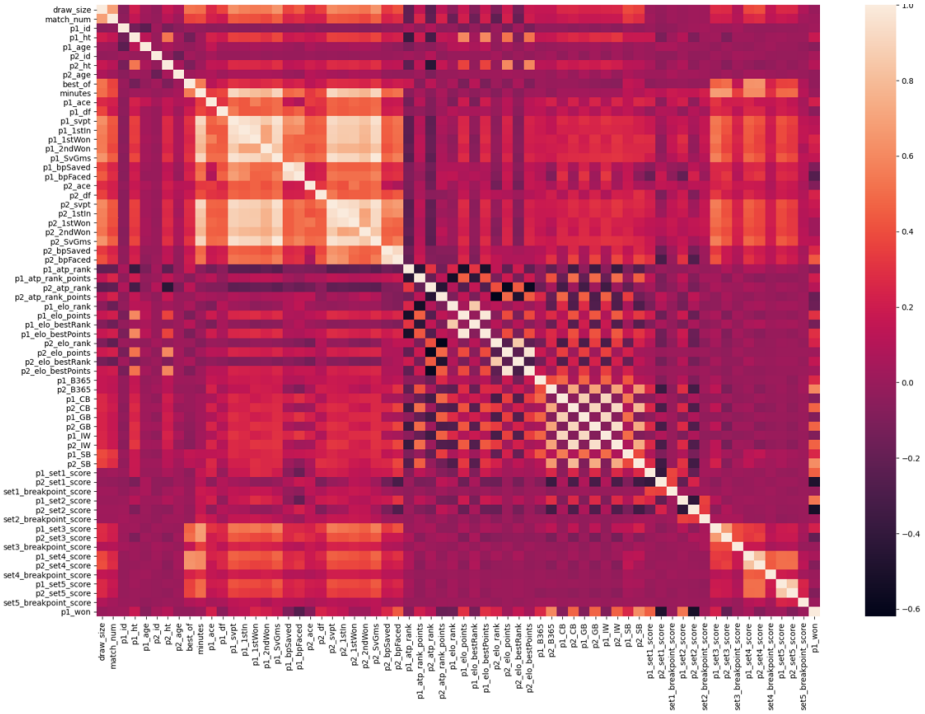
גרף 4: אחוז ערכים חסרים עבור קטגוריה ד

בגרף זה ישנם הרבה ערכים חסרים. למרות זאת נתייחס לאחר בדיקת קורלציות מה נעשה עם התכונות הללו.

בשביל לצמצם עוד את כמות התכונות, נבדוק האם יש קורלציה גבוהה בין חלק מהתכונות, בהתאם נחליט אלו תכונות להוריד ואלו להשאיר.

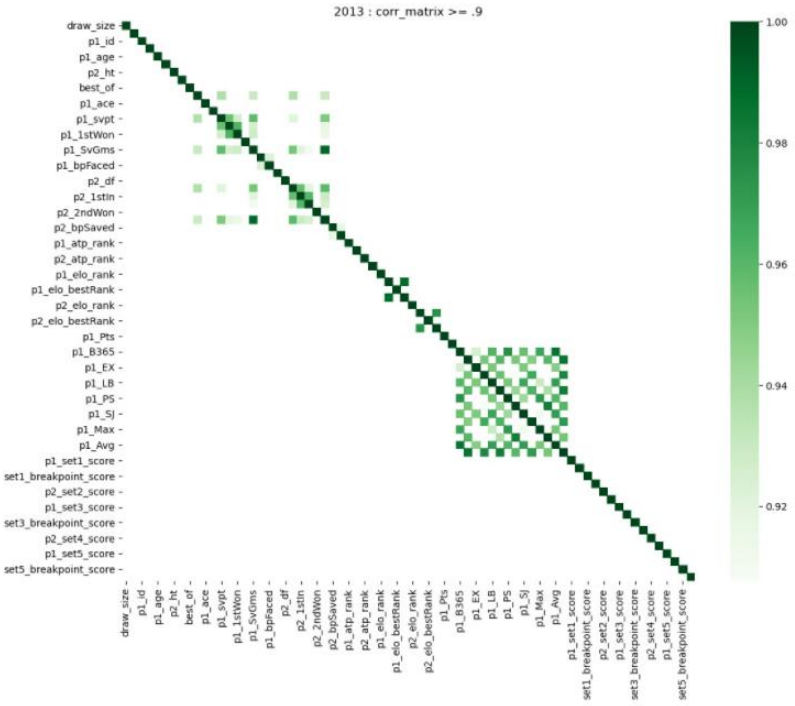
קורלציה- התאמה בין שני משתנים. ככל שההתאמה קרובה יותר ל-1 סימן שיש קשר גדול בין המשתנים.

נבדוק קורלציה בין התכונות שלנו:



גרף כ: טבלת קורלציה ראשונית

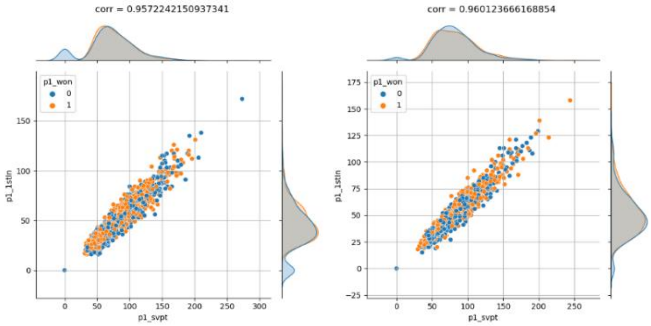
ניתן לראות כי הגרף הינו עמוס ולא פשוט לניתוח. לכן, ניצור גרף אשר יציג לנו קורלציות מעל 0.9:



גרף כ: טבלת קורלציה מרוכזת

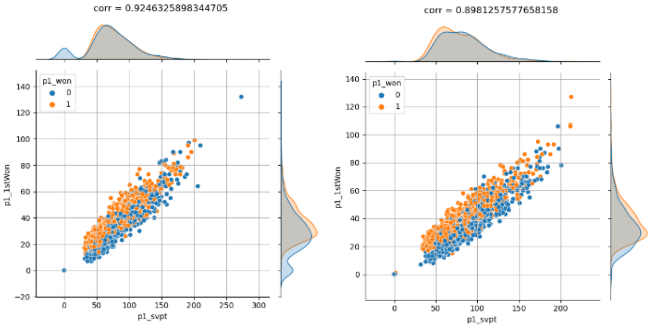
נבדוק קורלציות בין התכונות שנצבעו בצבע חזק בגרף ונדון האם ניתן לוותר על חלק מהתכונות הללו. אם הוחלט לוותר, ננמק ונסביר למה הגיוני שקיים קורלציה גבוהה בין התכונות.

בין התכונות p1_svpt, p1_1stIn קיימת קורלציה בממוצע של 0.95 בין השנים 2001-2021. ניתן להסביר זאת ע"י כך שרוב שחקני הטניס רוצים להימנע מהגשות שניות ולכן רוב ההגשות שנכנסות הינן הגשות ראשונות. ניתן לשים לב כי הדאטה שלנו באמת מראה שזהו המצב כיוון שה-svpt (כמות ההגשות בכללי) שווה כמעט לכמות ההגשות הנכנסות בהגשה הראשונה (stln1). לכן, נרצה לוותר על התכונה p1_1stIn.



גרף 7: דוגמה לקורלציות בין התכונות p1_1stIn, p1_svpt

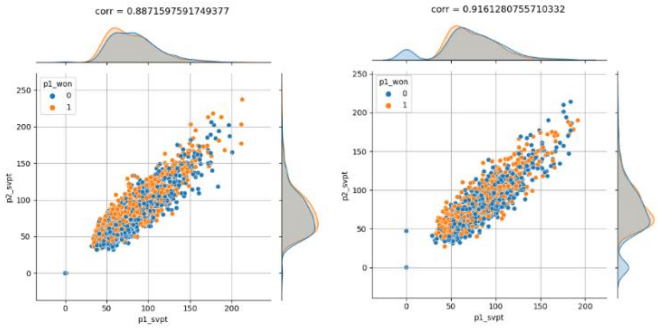
בין התכונות p1_svpt, p1_1stWon קיימת קורלציה בממוצע של 0.92. כמות ההגשות הראשונות כמעט זהה לכמות הניצחונות בהגשה הראשונה, וניתן לראות את היתרון של השחקן המגיש. במקרה זה, לא נרצה לוותר על אחת מהתכונות מכיוון שלאחר הסתכלות על הגרפים של תלות בין התכונות, נראה כי הם 'מחלקים' יפה את הדאטה לפי התיוג שלהם.



גרף 8: דוגמה לקורלציות בין התכונות p1_1stWon, p1_svpt

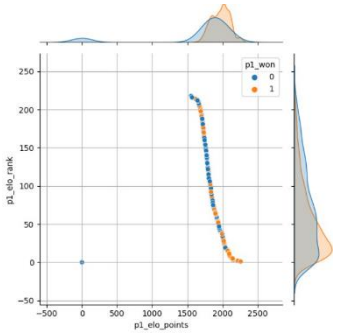
בין התכונות p1_svpt, p2_svpt קיים קורלציה בממוצע של 0.9. בכל משחקון מגיש אחד השחקנים לאורך כל המשחקון. כמות המשחקונים כמעט שווה, אך כמות הנקודות בכל משחקון לא בהכרח שווה, לכן הגרף אינו לינארי לגמרי (y=x), ויש משחקים בהם לשחקן אחד יותר נקודות בהן הוא מגיש, מאשר השחקן השני. למרות הקורלציה הגבוה לא נרצה למחוק את אחת מהתכונות הללו כיוון שאין באמת קשר בניהן והן שייכות לשחקנים שונים.

הורדה של אחת התכונות תגרום לאיבוד מידע חשוב ולכן לא כל פעם שתהיה לנו קורלציה גבוהה, בין שתי תכונות, עלינו לסמוך על זה ולמחוק את אחת מהן.



גרף 9: דוגמה לקורלציות בין התכונות: p1_svpt, p2_sv

בין התכונות pi_elo_rank, pi_elo_points קיימת קורלציה בממוצע של 0.5-.כלומר הקורלציה הינה הפוכה, וזאת מכיוון שככל שהדירוג נמוך יותר מספר הנקודות גבוהה יותר. לכן נמחק את התכונה pi_elo_points.



גרף 10: דוגמה לקורלציות בין התכונות: pi_elo_rank , pi_elo_points

בנוסף, ראינו כי בשנים שונות השתתפו בתי הימורים שונים, כאשר בחלק מהשנים יש יותר חפיפה ובחלק פחות. מצאנו כי רוב הערכות ההימורים בין בתי ההימורים השונים קורלטיביים.

בשביל להוריד את כמות התכונות (שאינה אחידה בין השנים) ובשביל לקבל תמונת מצב אחידה לכל השנים נרצה לאחד את כלל ההימורים עבור כל משחק לתכונה אחת. מכיוון שטווח הערכים עבור ההימורים שונה בין כל בית הימורים, נבצע נרמול z-score ולאחר מכן נחשב את הממוצע.

לסיכום, התחלנו עם 69 תכונות ונשארנו עם 60 תכונות.

עבור התכונות: tourney_name, tourney_id, match_num, Pi_name, pi_ioc

החלטנו שלא להשתמש בהן כיוון שלדעתנו אינן מועילות לנו (לא מוסיפות מידע חשוב). בנוסף עבור התכונות pi_ioc, tourney_name לא ניתן להפוך אותם למידע מספרי מהמחרוזת שממנה המידע הזה מיוצג ולכן רוב אלגוריתמי הלמידה שנשתמש בהם לא יכולים להשתמש בתכונות.

עבור התכונות: Court(25%), pi_seed(70%), pi_entry(80%)

חסר הרבה מידע כפי שציינו למעלה ועל כן החלטנו להוריד אותן.

עבור p1_1stIn: קיימת קורלציה גבוהה בין תכונה זו לתכונה p1_svpt ולכן כפי שהסברנו למעלה החלטנו להוריד אותה.

עבור pi_elo_points: קיימת קורלציה גבוהה בין תכונה זו לתכונה pi_elo_rank ולכן כפי שהסברנו למעלה החלטנו להוריד אותה.

הימורים - כפי שהסברנו קודם לקחנו את ההימורים ואיחדנו אותם לקטגוריה אחת.

לאחר צמצום כמות התכונות, נעבור לביצוע נרמול לתכנות. נדון לגבי שני סוגים של נרמול:

<u>נרמול min-max</u> – ננרמל כאשר ערכי התכונה בדידים, ערכים שעברו טרנספורמציה כלשהי.

<u>נרמול z-score</u> – ננרמל כאשר התפלגות הערכים דומה להתפלגות נורמלית (צורת פעמון), וכאשר

הערכים לא עברו טרנספורמציה כלשהי.

כפי שלמדנו, נרמול מאוד חשוב והכרחי לביצוע לפני הרצת אלגוריתמי למידה. ישנם אלגוריתמי למידה אשר משתמשים במרחק האוקלידי בין התכונות, לכן אם לא נבצע נרמול, טווח התכונה מאוד ישפיע על המרחק ועל הנכונות של האלגוריתמים.

עבור כל אחת מהתכונות הדפסנו ובדקנו את הגרף המראה את התפלגות הדאטה ולאחר מכן החלטנו איזה נרמול מתאים לאותו הגרף. כיוון שגרפים אלו תופסים הרבה מקום ולא מאוד מעניינים, לא נוסיף אותם.

מתודולוגיה ניסויית שאלות 1-3

עבור שלושת השאלתו הראשונות, נרצה למצוא את האלגוריתם ה Classification האופטימלי, לכן נכוונן את הפרמטרים שציינו לכל אלגוריתם. נעשה זאת בעזרת כיוונון דינאמי של פרמטרים.

לכל פרמטר שנרצה לכוונן, נבחר רשימת ערכים הגיוניים, ועבור כל ערך כזה נמצא את הממוצע המקסימלי של הדיוק בשיטת cross validation.

את שיטת cross validation בפרויקט שלנו נבצע ע"י חלוקה של הסט האימון ל-5 קבוצות, חיבור 4 קבוצות יחדיו ומדידת הדיוק עם הקבוצה החמישית.

נבצע חזרה על פעולה זו 5 פעמים כך שבכל פעם קבוצה אחרת מהווה את סט הוולידציה ודיוק הוולידציה של הפרמטרים יקבע לפי הממוצע של חמשת הריצות.

נרשום עבור כל אלגוריתם את התוצאה הכי טובה שקיבלנו. נדון על בחירת הפרמטרים (האם לבחור את הפרמטר שנתן את הדיוק הכי טוב או לבחור פרמטר אחר שלדעתנו יגיע לדיוק טוב יותר על קבוצת המבחן) ונבצע הרצה של הפרמטר הנבחר על קבוצת המבחן.

תיאור הניסויים, תוצאות ומסקנות שאלות 1-3

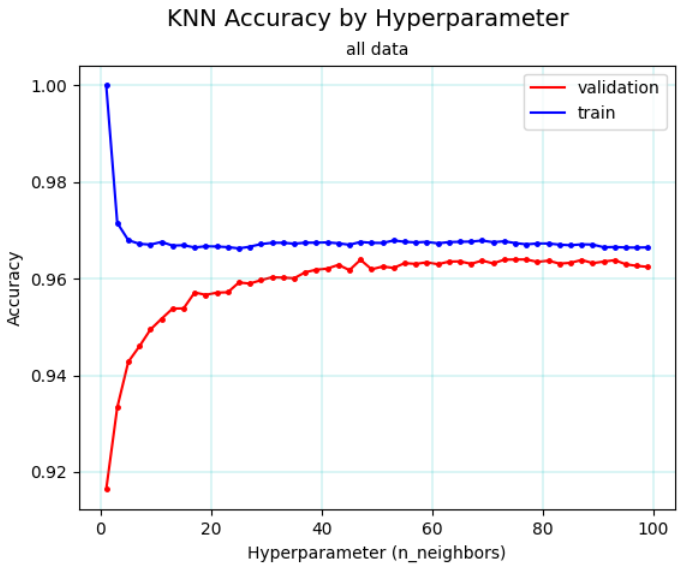
שאלה 1: בהינתן כל המידע שנצליח למצוא ברשת על משחק טניס בין שני שחקנים – מי ינצח.

כיוונון פרמטרים

<u>KNN:</u>

Validation Best k_neigh is: 75 with acc: 0.963

Train Best k_neigh is: 1 with acc: 1.0



גרף 11:כיוונון KNN שאלה 1

עבור קבוצת האימון נוכל לראות שעבור שכן אחד, יש דיוק של 100% זאת כיוון שמודל ה-KNN שומר את כל הדוגמאות מקבוצת האימון וכאשר אנו מבצעים predict על קבוצת האימון עצמה עם k=1, KNN מוצא לכל דוגמה את עצמה בתור השכן הכי קרוב (שכן יש להם את אותן ערכי תכונות) ומחזיר את התיוג של אותה הדוגמה, ולכן אין שגיאה כלל.

ככל שמסתכלים על יותר שכנים, כך יש יותר דוגמאות שלווא דווקא עם אותו התיוג וכך דיוק החיזוי יורד.

עבור קבוצת הוולידציה, הדוגמאות חדשות ולכן KNN לא מכיר אותן. עבור k קטן, הדיוק פחות טוב, אך ככל שעולים בכמות השכנים הדיוק משתפר עד שנשאר דיי יציב.

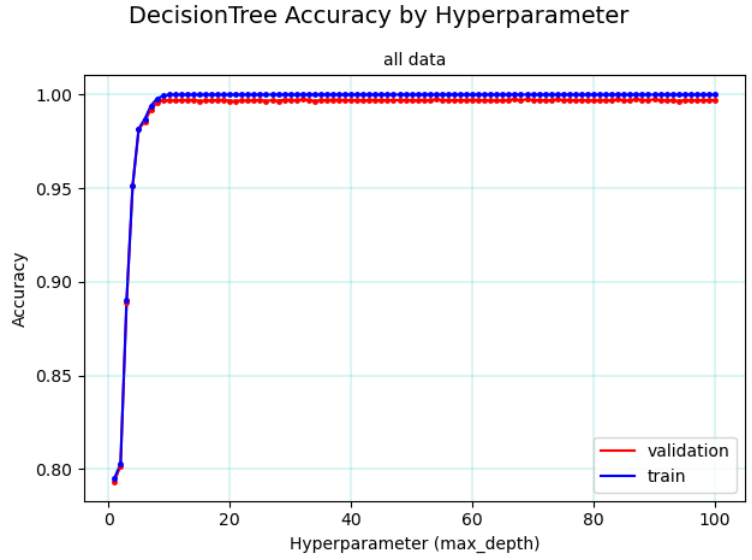
מהגרף נסיק כי עבור הדאטה שלנו כאשר קובעים לפי מס' מועט מידיי של שכנים, אנחנו במצב של underfitting שכן כל דוגמה דומה עם תיוג שונה, משפיעה המון על החיזוי.

ניתן לראות כי החל מ $k = 21$, הדיוק נשאר יציב. נסיק מכך כי מעל מספר זה רוב הדוגמאות הקרובות אלינו בעלות אותו סיווג, והוספה של מס' מועט של דוגמאות, לא ישנה את סיווג הKNN. בגלל כמות הדוגמאות(כ 47 אלף) נרצה דווקא K גדול מ 21 ולכן נבחן את ה K להיות דיוק הוולידציה הכי טוב שקיבלנו.

<u>Decision Tree:</u>

Validation Best max_depth: 87 with acc: 0.997

Train Best max_depth: 11 with acc: 1.0



גרף 12: כיוונון decision tree שאלה 1

מהגרף, ניתן לראות שהחל מעומק 10 עבור סט האימון וגם סט הוולידציה, הדיוק שואף ל-1.

עבור ה-train, מגיעים לדיוק 1 כאשר בונים עץ בעומק 11. החל מעומק זה נשארנו באותו הדיוק – האלגוריתם המשיך להעמיק את העץ, אך זה אינו דבר הכרחי, שכן כבר הגענו לדיוק המקסימלי. כפי שלמדנו, ההעמקה מעבר לעומק 11 תגרום ל overfitting.

עבור ה-validation, מצאנו כי העומק הטוב ביותר הוא 87. מהגרף, ניתן לראות כי הדיוקים כמעט זהים החל מעומק 11, לכן נבחר את עומק 11 עבור המודל שילמד, כדי להימנע מoverfitting.

עשרת התכונות המשמעותיות ביותר:

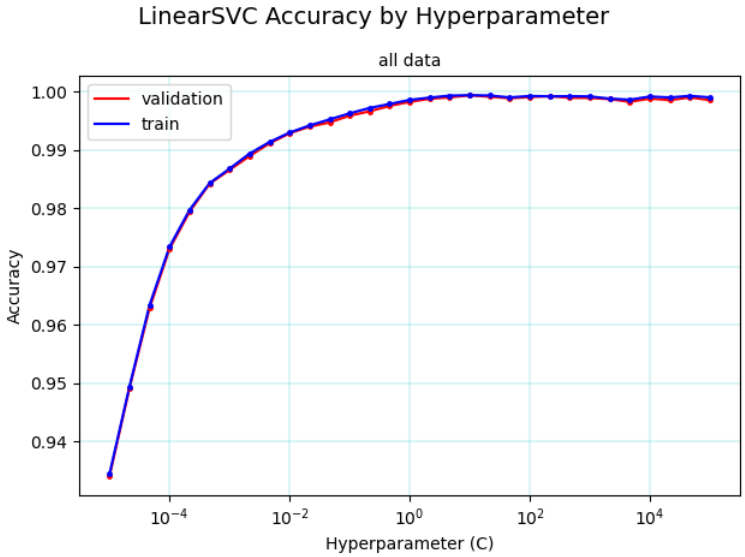
p2_betting_odds, p2_set2_score, p1_set2_score, p2_set3_score, p1_set3_score, p1_set5_score, p2_set5_score, best_of, p2_set4_score, p1_set4_score

דבר המעיד על כך שהאלגוריתם זיהה את התכונות של ניקוד המשחק בזמן אמת למשמעותיות ביותר ולכן קובעות את התיוג הסופי.

:SVM

Validation Best C: 10.0 with acc: 0.999

Train Best C: 10.0 with acc: 0.999



גרף 13: כיוונון SVM שאלה 1

מהגרף ניתן לראות כי עד ל $C = 1$ הדיוק עולה ככל שמקדם הרגולריזציה עולה. החל מ $C = 1$ הדיוק יציב עם הפרשים מינורים בין המקדמים השונים.

דבר זה מצביע על כך שעדיף מקדם רגולריזציה גבוה 'יחסית' כלומר לתת "עונש" גדול לטעויות סיווג שמוביל אותנו לאלגוריתם hard-svm, ולכן הגדלת ה-C אחרי $C = 1$ כמעט ולא משפיעה על הדיוק (כיוון שוקטור ה-w שנמצא יהיה זהה בכיוון ויפריד טוב את הדאטה ללא התייחסות ל-margin). מהדיוקים הגבוהים נסיק כי הדאטה שלנו כמעט פריד לינארית. (אם היה פריד לינארית היינו מקבלות דיוק 1)

הדיוק הכי גבוה שקיבלנו מהאלגוריתם שלנו לכוונון הפרמטרים התקבל עבור $C = 10$, אנו יודעות (מקורס מבוא למערכות לומדות) שזהו מקדם רגולריזציה לא גבוה מידיי שיכול לגרום ל-*overfitting* ולא נמוך מידיי שיכול לגרום ל-*underfitting*. ולכן נבחר בערך זה.

אנו משערות שניתן להפריד לפי תוצאות המשחקים בין שני השחקנים.

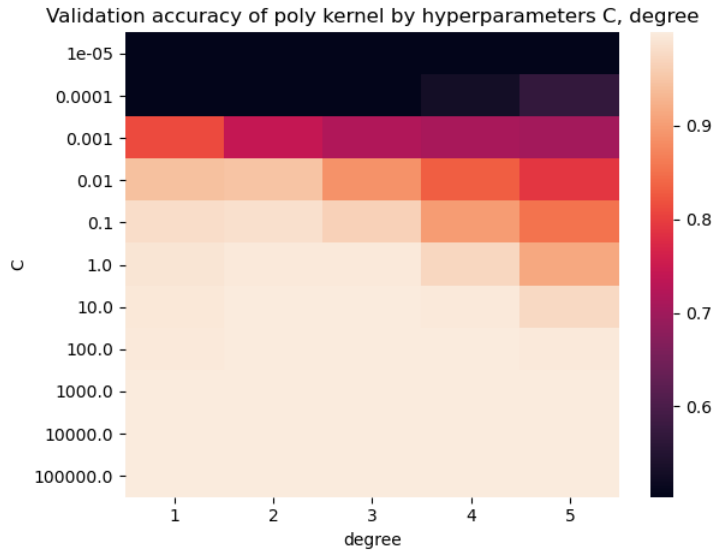
עשרת התכונות המשמעותיות ביותר:

p1_bpFaced, p2_bpFaced, p1_set5_score, p2_set5_score, p1_bpSaved, p2_bpSaved, p2_set4_score, p1_set4_score, p2_set3_score, p1_set3_score

ניתן להבחין כי התכונות ממושקלות בזוגות, כלומר האלגוריתם מפריד בין תכונות של p1 לתכונות של p2.

:SVM – poly kernel

Validation Best parameters: C=100000.0, degree=1 with accuracy: 0.9996



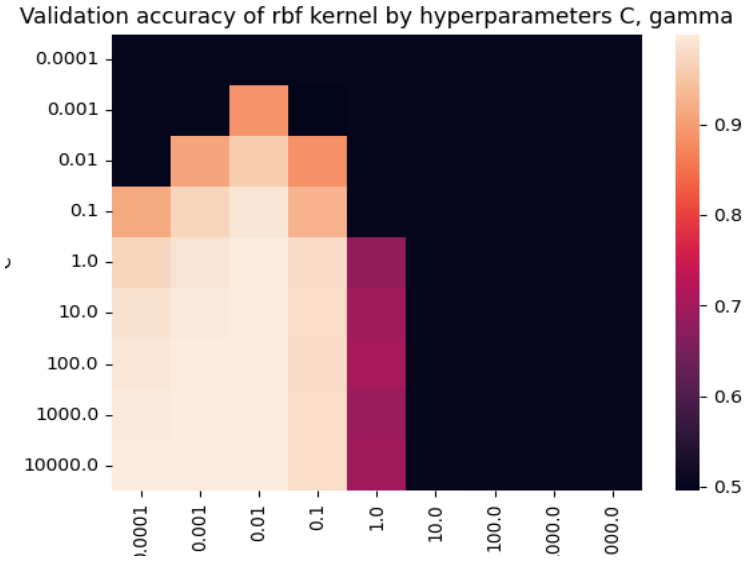
גרף 14: כיוונון SVM – poly kernel שאלה 1

זהו C מאוד גבוה שמעיד על overfitting, בנוסף, דרגה 1 מצביעה על SVM רגיל ללא קרנל פולינומי.

לכן נבחר זוג פרמטרים עם C קטן יותר אך עם דרגה שונה 1 כדי לבדוק יישום של קרנל פולינומי ועם דיוק ולידציה דומה (ניתן לראות בטבלת הדיוקים שיש המון זוגות פרמטרים עם דיוק גבוה מאוד). הפרמטרים שבחרנו הינם:C=10.0, degree=2 with accuracy: 0.9994

:SVM – rbf kernel

Validation Best parameters: C=1000.0, gamma=0.001 with accuracy: 0.9993



גרף 15: כיוונון SVM - rbf kernel שאלה 1

קיבלנו C גדול וgamma קטן עבור דיוק מקסימלי של הוולידציה, ולכן נבחר זוג פרמטרים אחרים, על מנת להימנע מ overfitting, עם דיוק דומה(0.9988): C=100, gamma=0.01.

Random Forest:

באלגוריתם זה היינו צריכות לכוונן מספר פרמטרים יחדיו. מכיוון שמס' האפשרויות עצום וכל הרצה כזו תיקח יותר מידיי זמן, בנינו רשימה של ערכים אפשריים לכל אחד מהפרמטרים שרצינו לכוונן ודגמנו באקראיות ערכים שונים מכל פרמטר. ביצענו זאת מס' גדול של איטרציות, וכך קיבלנו בהסתברות טובה את כיוונון הפרמטרים שלנו.

רשימת הערכים אותם כיוונו:

n_estimators: [100, 190, 280, 370, 460, 550, 640, 730, 820, 910, 1000]

max_features: [sqrt, log2]

max_depth: [1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, None]

min_samples_split: [2, 5, 10]

min_samples_leaf: [1, 2, 4]

criterion: [gini, entropy]

ביצענו 50 איטרציות של בחירה אקראית של הערכים הללו ומדידת הדיוק שלהם בעזרת cross-validation. קיבלנו את אוסף הפרמטרים הטובים ביותר הנ"ל: (עם דיוק של 0.998)

n_estimators: 370, min_samples_split: 5, min_samples_leaf: 1, max_features: sqrt, max_depth: 20, criterion: entropy

לאחר בחינת התוצאות בנינו רשימת טווחי ערכים מצומצמת יותר שעליה הרצנו cross-validation עם כל הפרמוטציות האפשריות של הפרמטרים. הרשימה שבחנו הינה:

max_depth: [10, 15, 20, 25, 30]

max_features: [sqrt]

min_samples_leaf: [1, 2, 3]

min_samples_split: [3, 5, 7]

n_estimators: [320, 370, 410]

criterion: [entropy]

התוצאה הכי טובה שקיבלנו היא עבור פרמטרים הנ"ל: (עם דיוק 0.9983)

n_estimators: 320

min_samples_split: 3

min_samples_leaf: 1

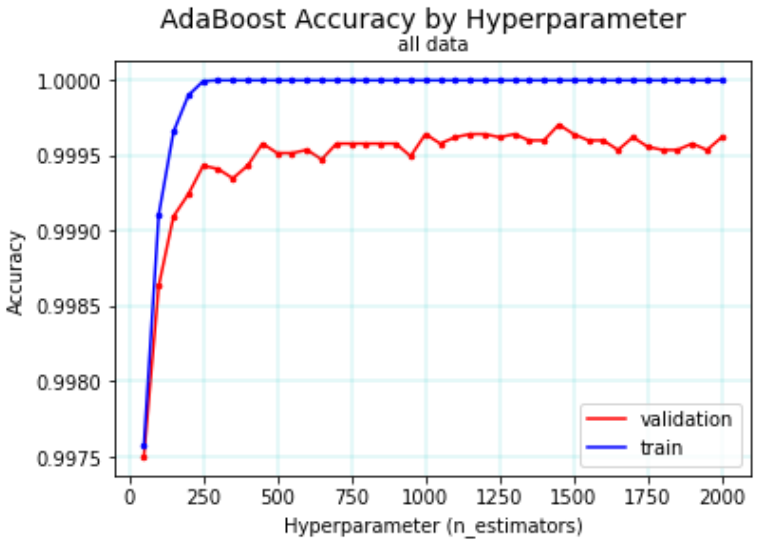
max_features: sqrt

max_depth: 15

criterion: entropy

:AdaBoost

Validation Best n_estimators: 1450 with acc: 0.9997
Train Best n_estimators: 300 with acc: 1.0



גרף 16: כיוונון AdaBoost שאלה 1

ניתן לראות שככל שיש יותר מסווגים חלשים (שורשי עצים) המסווג הכללי יותר 'עשיר' ומצליח לסווג יותר מקרי קצה.

החל מ-300 מסווגים חלשים הדיוק הינו מקסימלי עבור סט האימון והדיוק על סט הוולידציה נשאר יציב. נשים לב כי עבור סט האימון מס' זה מספיק על מנת להתאים את כל הדאטה.

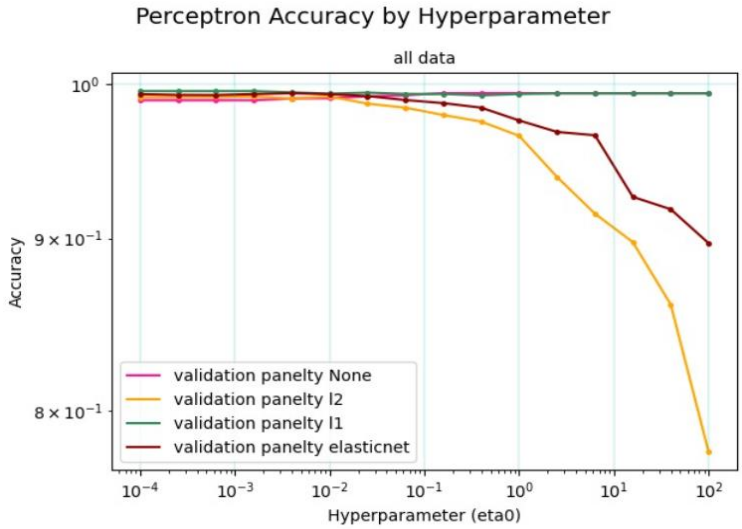
נבחר את ערך הפרמטר n_estimators=1250 (ולא 1450 כמו שקיבלנו עבור דיוק ולידציה מקסימלי). כיוון שניתן לראות בגרף כי בערך זה יש קפיצה משמעותית שלדעתנו מעידה על overfitting. ולכן נבחר ערך מעט קטן יותר כשהערכים מסביבו גם יציבים.

עשרת התכונות המשמעותיות ביותר:

set4_breakpoint_score, p1_set5_score, p1_set3_score, set2_breakpoint_score, p1_set1_score, p1_set2_score, set1_breakpoint_score, p1_set4_score, p2_elo_bestRank, set3_breakpoint_score

:Perceptron

Validation best penalty: l1, 'eta0': 0.0001 with acc: 0.9954



גרף 17: כיוונון Perceptron שאלה 1

ניתן לראות מהגרף כי בחלק מפונקציות הרגלוזציה אנו מפספסים את נקודת המינימום בGD כיוון שמקדם הלמידה גדול מידיי ולכן הדיוק בהם יורד. נבחר את ה'eta0' הכי טוב שהאלגוריתם מצא לנו שהוא קטן מספיק כך שהצעדים בו קטנים ואנו מתכנסים בעזרתו למינימום לא משנה באיזו פונקציית רגולרזציה נבחר. בנוסף בחרנו את L1 כפונקצית הרגולרזציה כיוון שהיא נותן את דיוק הכי טוב.

עשרת התכונות המשמעותיות ביותר:

p2_set5_score, p1_set5_score, p2_set4_score, p1_set4_score, p2_set3_score, p1_set3_score, p2_set1_score, p2_set2_score, grand_slams, p1_set1_score

:MLP

בדומה לאלגוריתם Random Forest היינו צריכות לכוונן מספר פרמטרים יחדיו. מכיוון שמס' האפשרויות עצום וכל הרצה כזו תיקח יותר מידיי זמן, בנינו רשימה של ערכים אפשריים לכל אחד מהפרמטרים שרצינו לכוונן ודגמנו באקראיות ערכים שונים מכל פרמטר. ביצענו זאת מס' גדול של איטרציות, וכך קיבלנו בהסתברות טובה את כיוונון הפרמטרים שלנו.

החלטנו לבחון בין שכבה אחת ל-3 שכבות נסתרות (שהן לא שכבת ה-input או ה-output), ובכל שכבה כמות שונה של נוירונים.

רשימת הערכים אותם כיוונו:

hidden_layer_sizes = [(64,), (128,), (128, 128), (128, 64, 128), (64, 64), (64, 64, 64), (64, 128, 64), (128, 64, 32)]

activation = [logistic, tanh, relu]

solver = [adam, sgd]

alpha = np.logspace(-5, 2, num=8)

max_iter = [200, 500, 1000, 1500]

לאחר בחירת 50 קבוצות אקראיות, קבוצת הפרמטרים שהביאו לדיוק הכי טוב (0.9996):

hidden_layer_sizes: (128,)

solver: adam

max_iter: 1000

alpha: 0.01

activation: relu

ולכן אותה נבחן.

סיכום כיוונון שאלה 1 והרצת האלגוריתמים עם הפרמטרים הנבחרים על קבוצת המבחן

Algorithm	Parameters chosen	Cross validation accuracy	Test accuracy
KNN	k_neigh: 75	0.963	0.9647
Decision Tree	max_depth: 11	0.997	0.9972
SVM	C: 10.0	0.999	0.9991
SVM – poly kernel	C=10.0 degree=2	0.9994	0.9995
SVM – rbf kernel	C=100.0 gamma=0.01	0.9988	0.9995
Random Forest	criterion: entropy max_depth: 15 max_features: sqrt min_samples_leaf: 1 min_samples_split: 3 n_estimators: 320	0.9983	0.9124
AdaBoost	n_estimators=1250	0.9996	0.9693
Perceptron	penalty: l1 'eta0': 0.0001	0. 9954	0.9488
MLP	solver: adam max_iter: 1000 hidden_layer_sizes:(128) alpha: 0.01 activation: relu	0.9996	0.9654

טבלה 1: סיכום כיוונון שאלה 1 והרצת האלגוריתמים עם הפרמטרים הנבחרים על קבוצת המבחן

דיון עבור טבלת הסיכום:

מטבלה זו ניתן לראות כי בחרנו פרמטרים טובים בתהליך כיוונון הפרמטרים, שכן כל הדיוקים על קבוצת המבחן הינם באותו סדר גודל של הדיוקים על קבוצת הוולידציה. התוצאה המקסימלית עבור דיוק קבוצת המבחן התקבלה באמצעות אלגוריתם SVM. זאת מכיוון שהאלגוריתם יוצר מסווג לפי היחס בין התכונות השונות ולכן ידע למשקל נכון את תכונות תוצאות המשחקונים במהלך משחק הטניס כהכי חשובות (דבר אשר מצביע בבירור על התוצאה הסופית - מי ניצח) ועל כן הוא קיבל את הדיוק הכי גבוה. אחריו ניתן לראות מהטבלה כי Decision Tree הגיע לדיוק גם כן גבוה עבור קבוצת המבחן. ראינו גם כאן ש Decision Tree נותן חשיבות רבה מאוד לתכונות של תוצאות המשחקונים אך אלגוריתם זה בונה עץ לפי ערכים של תכונות בפני עצמם ללא התייחסות ליחס בין התכונות ולכן הדיוק מעט נמוך יותר.

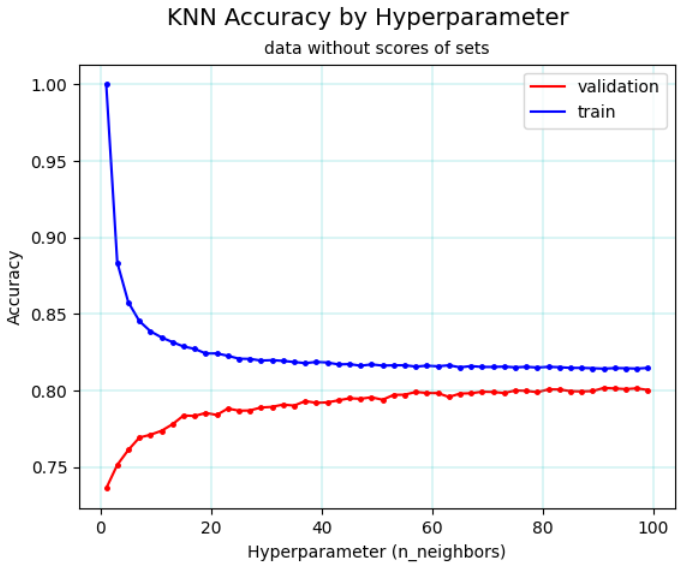
במבט כללי על הטבלה, כלל האלגוריתמים הניבו תוצאות טובות דבר שציפינו לו כיוון שחלק מתכונות הדאטה מכילות תוצאות זמן אמת על המשחק שמעידות על מי ניצח.

שאלה מס' 2: חיזוי תוצאות משחק בין שני שחקנים על סמך יכולות וביצועי השחקנים במהלך המשחק

כיוונון פרמטרים

KNN:

Validation Best k_neigh is: 91 with acc: 0.801
Train Best k_neigh is: 1 with acc: 1.0

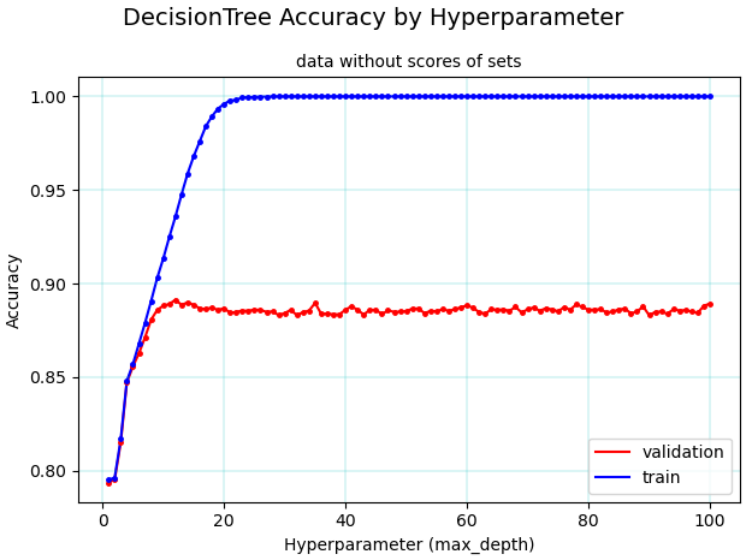


גרף 18: כיוונון KNN שאלה 2

ניתן לראות מהגרף מגמות דומות לתוצאות ה-KNN מהשאלה הקודמת, ככל שעולים במס' השכנים קיימת ירידה בדיוק של האימון ועלייה בדיוק הוולידציה. לכן נבחן את K=91.

Decision Tree:

Validation Best max_depth: 12 with acc: 0.891
Train Best max_depth: 29 with acc: 1.0



גרף 19: כיוונון decision tree שאלה 2

מהגרף ניתן לראות כי עד לעומק 12 שני הדיוקים (אימון וולידציה) עולים, והחל ממנו הדיוק של האימון ממשיך לעלות ודיוק הוולידציה 'נתקע' על 0.89 ויורד באיטיות.

הדיוק על האימון ממשיך לעלות כיוון שהעץ שנבנה בעל עומק גדול יותר כך שיש יותר הסתעפויות שמתאימות את עצמן לדאטה של האימון. הסתעפויות אלו גורמות ל-overfitting אשר גורם לדיוק הוולידציה לרדת.

נבחן את הפרמטר $max\_depth = 12$ שאותו קיבלנו מאלגוריתם כיוונון הפרמטרים שלנו.

הפיצ'רים המרכזיים בעץ בעומק 12 הם:

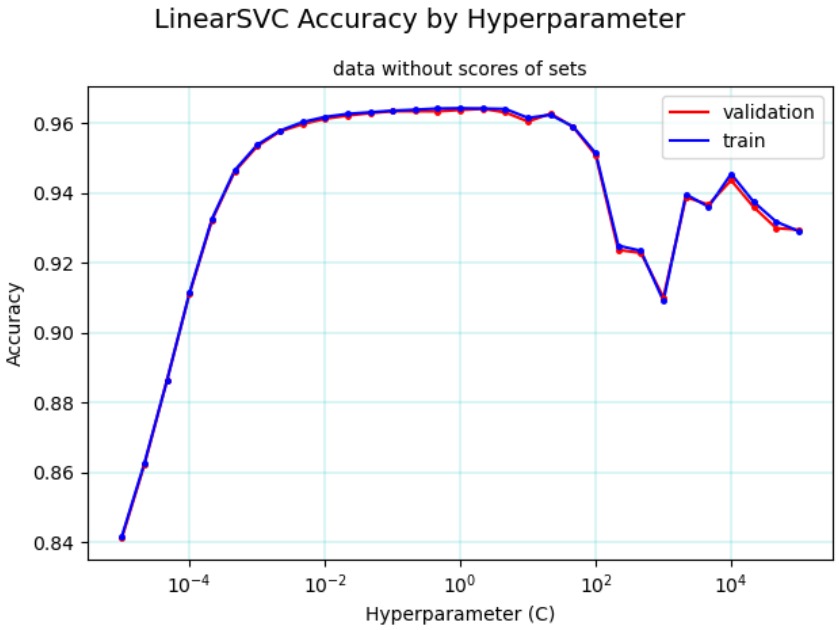
p2_betting_odds, p2_bpFaced, p1_bpFaced, p1_1stWon, p2_1stWon, p1_betting_odds, p2_bpSaved, p1_bpSaved, p1_2ndWon, p2_2ndWon

תכונות אלה הינם על ביצועי השחקנים בזמן המשחק וסטטיסטיקות שמצביעות על מי מנצח לפי בית הימורים. לכן נסיק שסה"כ ההימורים מצביעים על מי ינצח .

<u>:SVM</u>

Validation Best C: 2.1544 with acc: 0.964

Train Best C: 1.0 with acc: 0.964



גרף 20: כיוונון SVM שאלה 2

מגרף ניתן לראות עלייה בדיוק עד ל $C = 0.01$ ממנו התייצבות, והחל מ $C = 100$ ירידה.

כאשר מקדם הרגולריזציה נמוך (מתחת ל $C = 0.01$), ניתן דגש יותר גדול על יצירת margin מאשר על הפרדה של סט האימון ע"י w, (העונש על טעויות קטן). הדיוקים הנמוכים עבור מקדמים אלו, מצביעים על כמות טעויות גדולות יותר עבור הדאטה שלנו.

כאשר מקדם הרגולריזציה גבוה (מעל $C = 100$), ניתן לראות מהגרף שהדיוקים יורדים, כיוון שיש שאיפה ליצור מסווג לינארי אך דבר זה אינו אפשרי מה שמגדיל את מספר הטעויות ופוגע בדיוק.

מכיוון שהדיוק אינו 1, נסיק כי הדאטה שלנו אינו פריד לינארית.

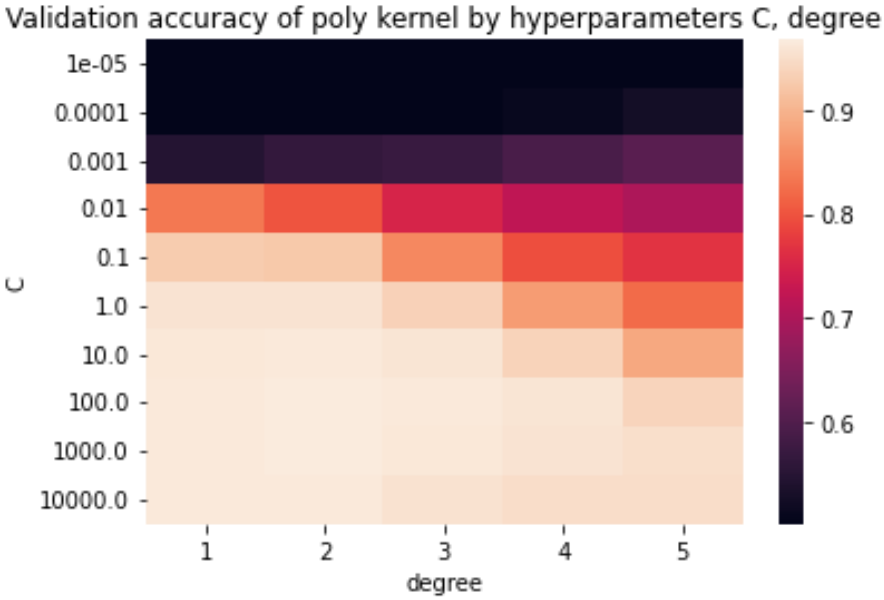
נבדוק אילו תכונות הן הדומיננטיות ביותר שמודל SVM כזה לומד. שמונת התכונות הינן:

p1_bpFaced, p2_bpFaced, p2_1stWon, p1_bpSaved, p1_1stWon, p2_svpt, p1_svpt, p2_bpSaved, p2_2ndWon, p1_2ndWon

גם כאן האלגוריתם שם דגש על תכונות בזוגות של p1, p2 (באופן פחות מובהק מהשאלה הקודמת).

:SVM – poly kernel

Validation Best parameters: C=1000.0, degree=2 with accuracy: 0.9668

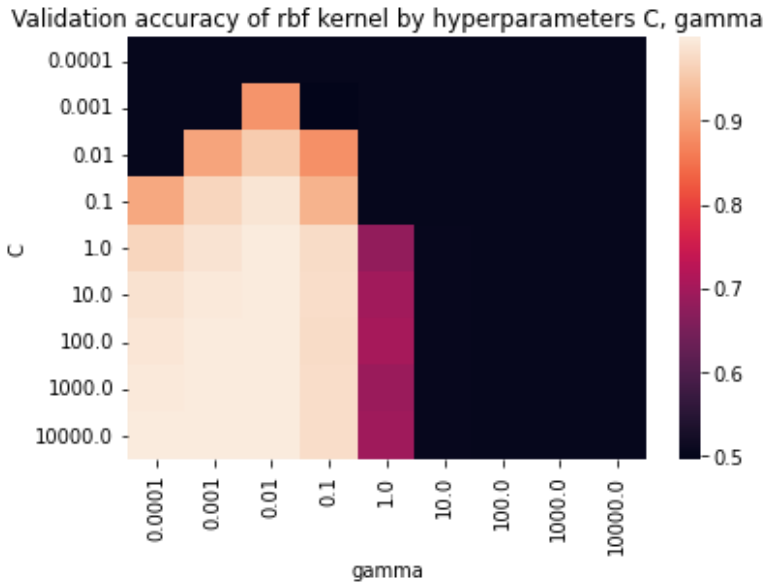


גרף 21: כיוונון SVM – poly kernel שאלה 2

מקדם הרגולריזציה שקיבלנו דיי גבוה מה שמעיד על overfitting ולכן נבחר C שקטן ב-2 סדרי גודל אך עם דיוק דומה (0.964): C=10.0, degree=2.

:SVM – rbf kernel

Validation Best parameters: C=10000, gamma=0.0001 with accuracy: 0.9658



גרף 22: כיוונון SVM - rbf kernel שאלה 2

קיבלנו C גדול וgamma קטן עבור דיוק מקסימלי של הווליידציה, ולכן נבחר זוג פרמטרים אחרים, על מנת להימנע מ overfitting, עם דיוק דומה(0.9653): C=10, gamma=0.01.

Random Forest:

ביצענו כיוונון פרמטרים בדומה לשאלה הקודמת שבחנו (דגימה הסתברותית מתוך אוסף ערכי פרמטרים), ובחרנו ברשימת ערכים זהה לזו של השאלה הקודמת.

לאחר ביצוע 50 איטרציות קיבלנו את אוסף הפרמטרים הטובים ביותר הנ"ל: (עם דיוק של 0.921)

n_estimators: 820, min_samples_split: 2, min_samples_leaf: 1, max_features: sqrt, max_depth: 40, criterion: entropy

לאחר בחינת התוצאות בנינו רשימת טווחי ערכים מצומצמת יותר שעליה הרצנו cross validation על כל הפרמוטציות האפשריות של הפרמטרים. הרשימה שבחנו הינה:

max_depth: [30, 35, 40, 45, 50]

max_features': [sqrt]

min_samples_leaf: [1, 2]

min_samples_split: [2, 3]

n_estimators: [770, 820, 870]

criterion: [entropy]

התוצאה הכי טובה שקיבלנו היא עבור פרמטרים הנ"ל: (עם דיוק 0.9208)

n_estimators: 820

min_samples_split: 2

min_samples_leaf: 1

max_features: sqrt

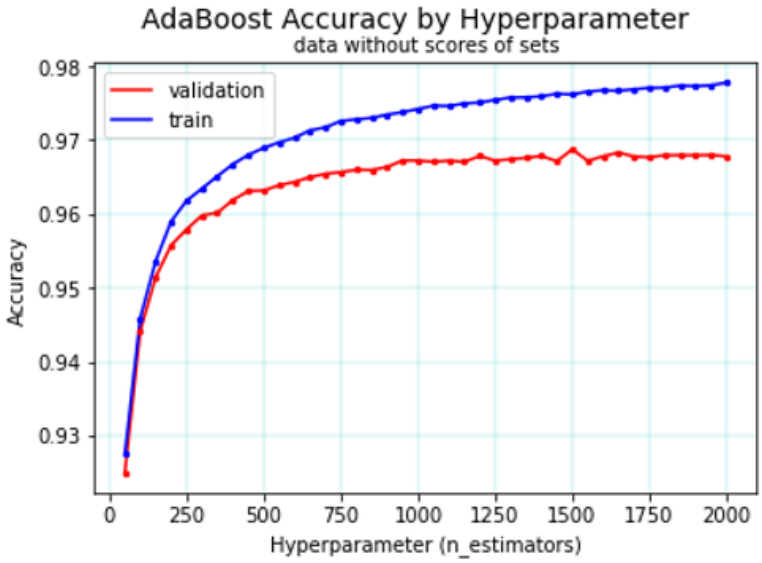
max_depth: 50

criterion: entropy

נשים לב כי לאחר הבדיקה הרנדומלית קיבלנו דיוק גבוה יותר מהדיוק אשר קיבלנו בהרצה של הפרמטרים הספציפיים וזאת בעקבות בחירת שונה (רנדומלית) של החלוקה לסט ולידציה וסט אימון. לכן הגיוני כי הדיוקים שונים קצת אך לא בסדר גודל משמעותי.

:AdaBoost

Validation Best n_estimators: 1500 with acc: 0.968
Train Best n_estimators: 2000 with acc: 0.977



גרף 23: כיוונון AdaBoost שאלה 2

בדומה לתוצאות השאלה הקודמת, ניתן לראות מהגרף שככל שיש יותר מסווגים חלשים (שורשי עצים) המסווג בעל דיוק גבוה יותר.

החל מ-500 מסווגים חלשים הדיוק עבור סט האימון ממשיך לעלות אך בצורה יותר מתונה, והדיוק עבור סט הוולידציה מתחיל להתאזן (עם קפיצות קלות). לכן נבחר לבחון את ערך הפרמטר n_estimators: 1500 שעבורו קיבלנו את דיוק הוולידציה הגבוה ביותר.

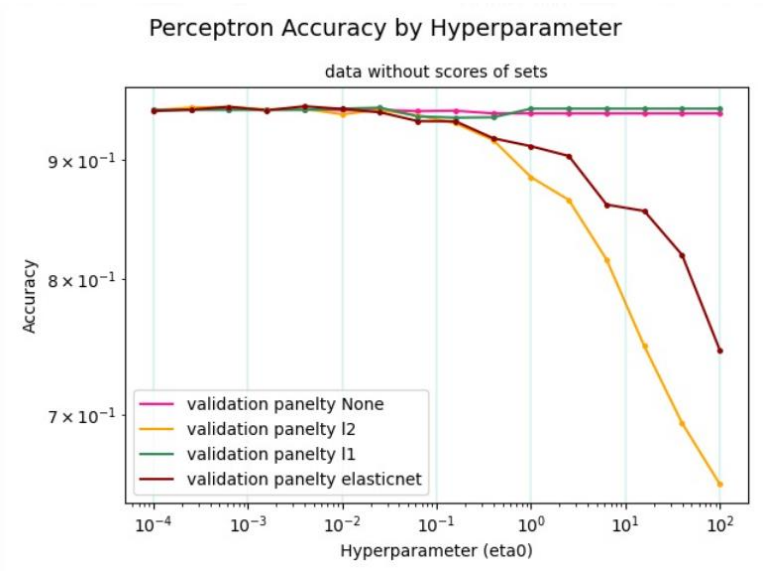
עשרת הפיצ'רים המשמעותיים ביותר:

p2_svpt, p1_svpt, p1_df, p2_df, p1_betting_odds, p2_elo_bestRank, p1_1stWon, p2_1stWon, p2_bpSaved, p1_bpSaved

גם כאן ניתן לראות כי האלגוריתם בחר כפרמטרים החשובים ביותר בתכונות אשר מצביעות על ביצועי השחקנים בזמן המשחק, דבר אשר מבציע באופן ישיר על מי ינצח.

:Perceptron

Validation best penalty: l2, etha0: $6 \cdot 10^{-4}$ with acc: 0.948



גרף 24: כיוונון perceptron שאלה 2

ניתן לראות מהגרף כי בחלק מפונקציות הרגלוזציה אנו מפספסים את נקודת המינימום ב-GD כיוון שמקדם הלמידה גדול מידיי ולכן הדיוק בהם יורד. נבחר את ה'eta0' הכי טוב שהאלגוריתם מצא לנו שהוא קטן מספיק כך שהצעדים בו קטנים ואנו מתכנסים בעזרתו למינימום לא משנה באיזו פונקציית רגולרזציה נבחר. בנוסף בחרנו את L2 כפונקצית הרגולרזציה כיוון שהיא נותן את דיוק הכי טוב.

תכונות משמעותיות ביותר:

p2_1stWon, p1_1stWon, 2_svpt, p1_svpt, p2_2ndWon, p1_bpFaced, p2_bpFaced, p1_2ndWon, p1_bpSaved, p2_betting_odds

:MLP

ביצענו כיוונון פרמטרים בדומה לשאלה הקודמת שבחנו (דגימה הסתברותית מתוך אוסף ערכי פרמטרים), ובחרנו ברשימת ערכים זהה לזו של השאלה הקודמת.

לאחר בחירת 50 קבוצות אקראיות, קבוצת הפרמטרים שהביאו לדיוק הכי טוב (0.9666):

solver: sgd

max_iter: 1000

hidden_layer_sizes: (128, 128)

alpha: 1.0

activation: tanh

ולכן נבחן פרמטרים אלו.

סיכום כיוונון שאלה 2 והרצת האלגוריתמים עם הפרמטרים הנבחרים על קבוצת המבחן

Algorithm	Parameters chosen	Cross validation accuracy	Test accuracy
KNN	k_neigh: 91	0.801	0.804
Decision Tree	max_depth: 12	0.891	0.891
SVM	C: 2.154	0.964	0.963
SVM – poly kernel	C: 10.0, degree: 2	0.964	0.9643
SVM – rbf kernel	C: 10, gamma: 0.01	0.9653	0.9655
Random Forest	criterion: entropy max_depth: 50 max_features: sqrt min_samples_leaf: 1 min_samples_split: 2 n_estimators: 820	0.9208	0.925
AdaBoost	n_estimators: 1500	0.968	0.969
Perceptron	penalty: l2 etha0: $6 \cdot 10^{-4}$	0.948	0.933
MLP	solver: sgd max_iter: 1000 hidden_layer_sizes: (128, 128) alpha: 1 activation: tanh	0.9666	0.964

טבלה 2: סיכום כיוונון שאלה 2 והרצת האלגוריתמים עם הפרמטרים הנבחרים על קבוצת המבחן

דיון עבור טבלת הסיכום:

מטבלה זו ניתן לראות כי בחרנו פרמטרים טובים בתהליך כיוונון הפרמטרים, שכן כל הדיוקים על קבוצת המבחן הינם באותו סדר גודל של הדיוקים על קבוצת הוולידציה.

לדעתנו לעומת השאלה הקודמת, הדאטה שלנו פחות פריד לינארית ולכן אלגוריתמי SVM סיפקו דיוק פחות טוב מאלגוריתם AdaBoost.

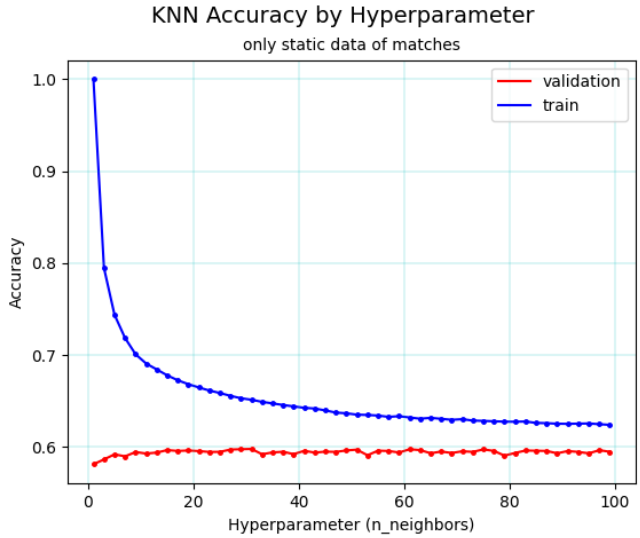
כלל האלגוריתמים הניבו תוצאות טובות. כלומר האלגוריתמים ידעו לתת חשיבות גבוהה לתכונות ביצועים של השחקנים בזמן המשחק ובכך ניבו טוב את התוצאה. כמצופה ככל שהאלגוריתמים יותר מורכבים אזי דיוקם גבוה יותר. KNN הביא את הדיוק הנמוך ביותר כיוון שזהו האלגוריתם הכי בסיסי מכלל האלגוריתמים שבחנו ומעט מעליו אלגוריתם Decision Tree שהביא לדיוק קצת יותר גבוה אך עדיין לא עובר את רף ה-90% דיוק.

שאלה מס' 3: חיזוי תוצאות משחק בין שני שחקנים על סמך נתונים התחלתיים של המשחק

כיוונון פרמטרים

KNN:

Validation Best k_neigh is: 39 with acc: 0.634
Train Best k_neigh is: 1 with acc:1.0



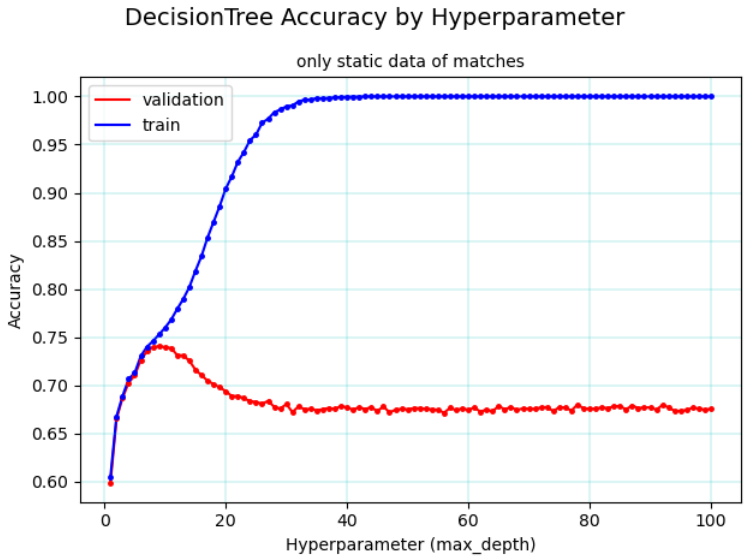
גרף 25: כיוונון KNN שאלה 3

עבור סט האימון המגמה זהה לשאלות הקודמות (ככל שעולים בכמות השכנים הדיוק יורד).

עבור סט הוולידציה, ניתן לראות מהגרף כי הדיוק כמעט ולא משתנה, לכן נסיק כי על סמך נתונים קרובים של משחקים אחרים, קשה לזהות מי הולך לנצח.

Decision Tree

Validation Best max_depth: 9 with acc: 0.74
Train Best max_depth: 44 with acc: 1.0



גרף 26: כיוונון decision tree שאלה 3

ניתן לראות בגרף תופעה דומה לדיון שלנו על עצי החלטה מהשאלה הקודמת.

עד עומק 9, יש עלייה של שני הדיוקים. החל ממנו עלייה בדיוק עבור האימון וירידה (משמעותית יותר מהשאלה הקודמת) בדיוק עבור הוולידציה - מצב של overfitting.

הפיצ'רים המרכזיים עבור עץ בעומק 9:

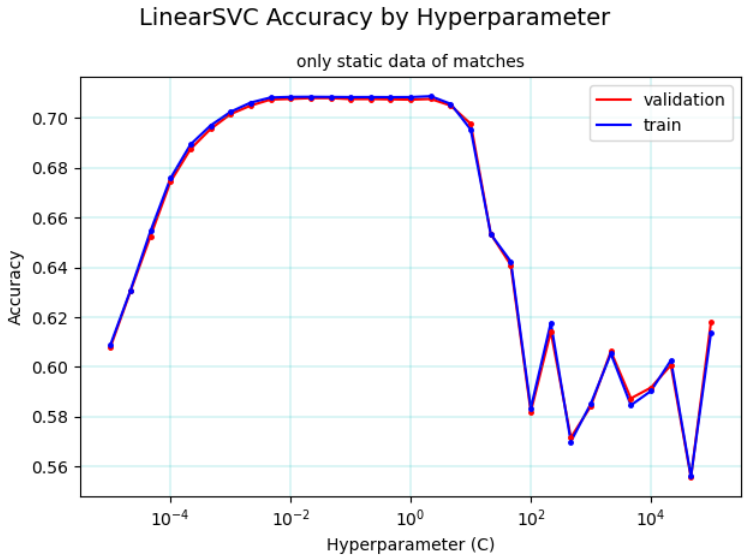
p2_elo_rank, p1_elo_rank, p1_elo_bestRank, p1_ht, p2_ht, p1_atp_rank_points, p2_atp_rank_points, p1_id, p1_age, tourney_date

ניתן לראות מבחירת התכונות כי האלגוריתם מסתמך בעיקר על הדירוגים הנוכחיים של השחקנים.

<u>SVM:</u>

Validation Best C: 0.046 with acc: 0.708

Train Best C: 2.154 with acc: 0.708



גרף 27: כיוונון SVM שאלה 3

בגרף ניתן לראות תופעה דומה לזו משאלה קודמת עבור אלגוריתם זה.

כאשר מקדם הרגולריזציה נמוך (מתחת ל $C = 0.005$), ניתן דגש יותר גדול על יצירת margin מאשר על הפרדה של סט האימון ע"י w, (העונש על טעויות קטן). הדיוקים הנמוכים עבור מקדמים אלו, מצביעים על כמות טעויות גדולות יותר עבור הדאטה שלנו.

עבור ערכים $0.005 < C < 10$ קיימת התייצבות.

כאשר מקדם הרגולריזציה גבוה (מעל $C = 10$), ניתן לראות מהגרף שהדיוקים יורדים ולא יציבים, כיוון שיש שאיפה ליצור מסווג לינארי אך דבר זה אינו אפשרי מה שמגדיל את מספר הטעויות ופוגע בדיוק. מכיוון שהדיוק אינו 1, נסיק כי הדאטה שלנו אינו פריד לינארית.

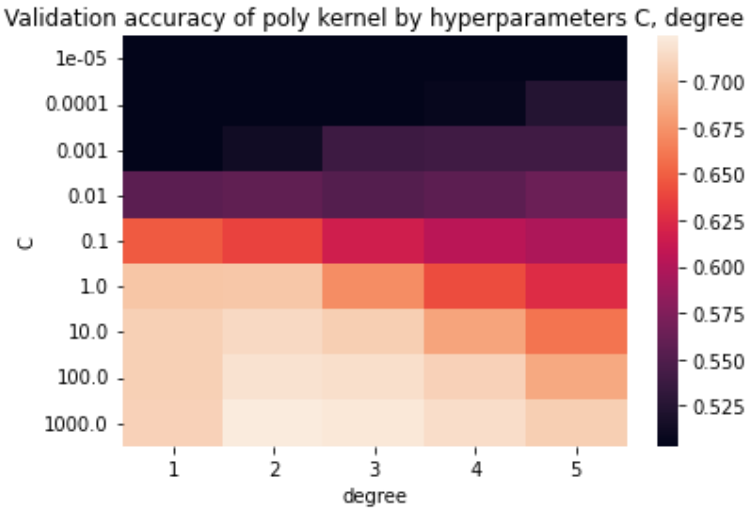
שמונת הפיצ'רים המשמעותיים ביותר:

p2_atp_rank, p1_atp_rank_points, p2_atp_rank_points, p2_elo_rank, p1_elo_rank, p1_elo_bestRank, p1_atp_rank, p1_ht

כאן המפריד פחות שם דגש על תכונות זהות של שני השחקנים, והדגש יותר על ניקודים ודירוגים כלליים של השחקנים.

:SVM – poly kernel

Validation Best parameters: C=1000.0, degree=2 with accuracy: 0.724

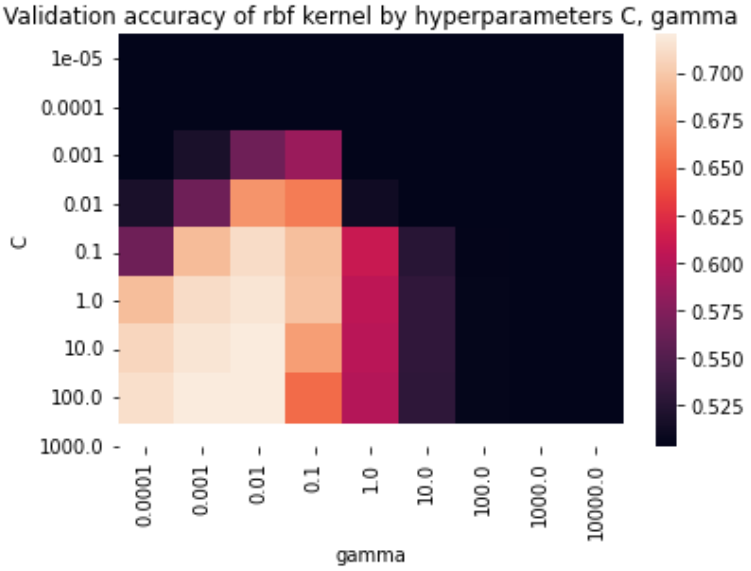


גרף 28: כיוונון SVM – poly kernel שאלה 3

מקדם רגולריזציה שקיבלנו דיי גבוה מה שמעיד על overfitting ולכן נבחר *C* שקטן ב-2 סדרי גודל אך עם דיוק דומה (0.713) : C=10.0, degree =2.

:SVM – rbf kernel

Validation Best parameters: C=100.0, gamma=0.01 with accuracy: 0.7201



גרף 29: כיוונון SVM - rbf kernel שאלה 3

קיבלנו C גדול עבור דיוק מקסימלי של הווליידציה, ולכן נבחר C שקטן בסדר גודל אחד, על מנת להימנע מ overfitting, אך עם דיוק דומה (0.716): C=10, gamma=0.01.

Random Forest:

ביצענו כיוונון פרמטרים בדומה לשאלה הקודמת שבחנו (דגימה הסתברותית מתוך אוסף ערכי פרמטרים), ובחרנו ברשימת ערכים זהה לזו של השאלה הקודמת.

לאחר ביצוע 50 איטרציות קיבלנו את אוסף הפרמטרים הטובים ביותר הנ"ל: (עם דיוק של 0.7463)

n_estimators: 460, min_samples_split: 10, min_samples_leaf: 1, max_features: sqrt, max_depth: 30, criterion: gini

לאחר בחינת התוצאות בנינו רשימת טווחי ערכים מצומצמת יותר שעליה הרצנו cross validation על כל הפרמוטציות האפשריות של הפרמטרים. הרשימה שבחנו הינה:

max_depth: [20, 25, 30, 35, 40]

max_features: [sqrt]

min_samples_leaf: [1, 2, 4]

min_samples_split: [2, 10, 12]

n_estimators: [350, 400, 450, 500]

criterion: [gini]

התוצאה הכי טובה שקיבלנו היא עבור פרמטרים הנ"ל: (עם דיוק 0.745).

n_estimators: 350

min_samples_split: 12

min_samples_leaf: 2

max_features: sqrt

max_depth: 40

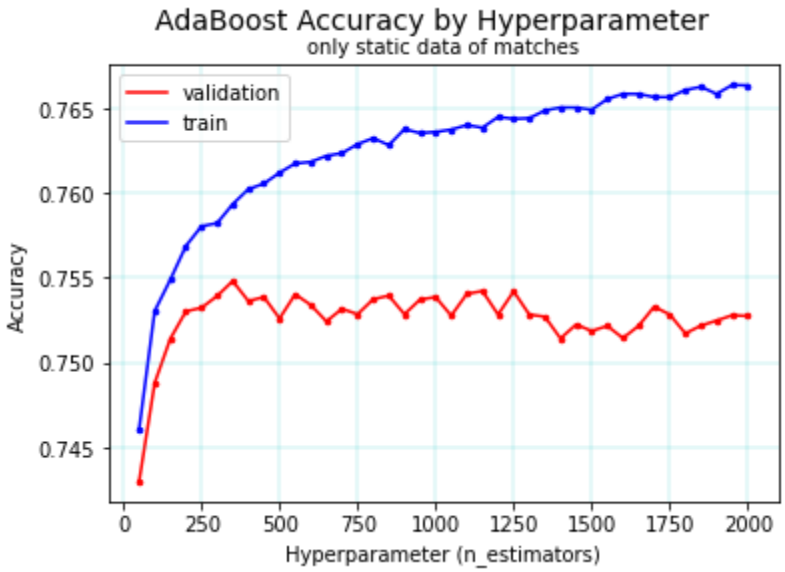
criterion: gini

נשים לב כי לאחר הבדיקה הרנדומלית קיבלנו דיוק גבוה יותר מהדיוק אשר קיבלנו בהרצה של הפרמטרים הספציפיים וזאת בעקבות בחירת שונה (רנדומלית) של החלוקה לסט ולידציה וסט אימון. לכן הגיוני כי הדיוקים שונים קצת אך לא סדר גודל משמעותי.

:AdaBoost

Validation Best n_estimators: 350 with acc: 0.754

Train Best n_estimators: 1950 with acc: 0.766



גרף 30: כיוונון AdaBoost שאלה 3

עבור האימון, יש עלייה מתמדת בדיוק ככל שכמות המסווגים החלשים עולה, מגמה אשר דומה לשאלות הקודמות עבור אלגוריתם זה.

עבור הוולידציה, החל מ-350 מסווגים יש ירידה איטית בדיוק, דבר המעיד על overfitting. לכן נבחר את הפרמטר n_estimators להיות 350.

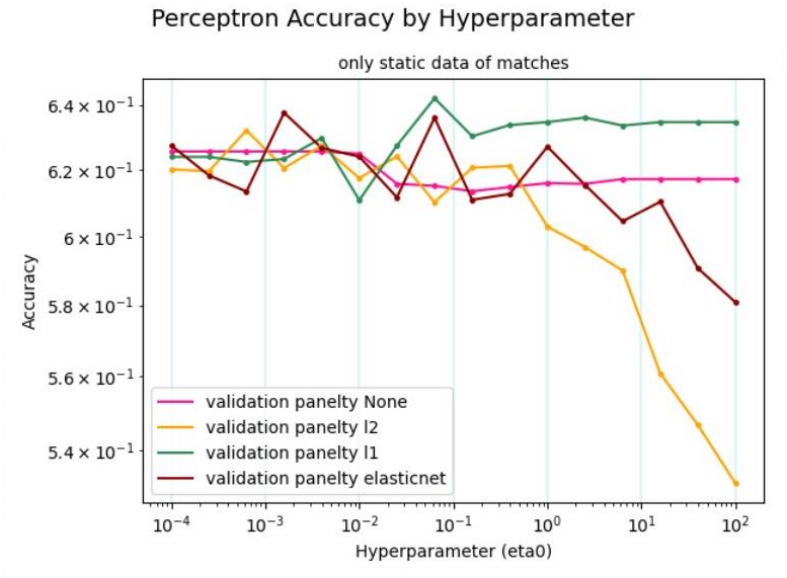
עשרת התכונות המשמעותיות ביותר:

p1_elo_bestRank, p1_hand, p2_elo_rank, p2_atp_rank_points, p2_hand, p2_atp_rank, p1_ht, tourney_date, p1_atp_rank_points, p1_atp_rank

תכונות אלה מצביעות על הדירוג של השחקנים ולכן הגיוני שהן נבחרו כמשמעותיות ביותר.

:Perceptron

Validation best penalty: l1, 'eta0': 0.063 with acc: 0.642



גרף 31: כיוונון perceptron שאלה 3

עבור פונקציית רגלוזציה L2 החל מ $eta0 > 1$ ישנה ירידה חדה בדיוק וזאת מכיוון שמקדם הלמידה גדול מידיי מכיוון שנקודת המינימום ב-GD מתפספסת. נשים לב כי בכלל, הדיוק עברו אלגוריתם זה לא מוצלח מכיוון שככל הנראה הדאטה שלנו מאוד לא פריד לינארי.

ניתן לראות מהגרף כי פונקצית השגיאה הטובה ביותר לסט הדוגמאות היא L1, עם מקדם למידה 0.063, שהוא קטן מספיק על מנת להגיע להתכנסות.

עשרת התכונות המשמעותיות ביותר:

p2_atp_rank, p2_atp_rank_points, p1_elo_bestRank, other_tour-level, p1_atp_rank_points, p1_elo_rank, p2_id, p2_elo_rank, p2_ht, grass

להפתעתנו, אלגוריתם זה בחר בתכונות שאמורות להצביע על מי ניצח (כגון דירוג השחקן) אך למרות זאת לא חזה טוב במיוחד (דיוק נמוך) לעומת אלגוריתמים קודמים שבחנו.

:MLP

ביצענו כיוונון פרמטרים בדומה לשאלה הקודמת שבחנו (דגימה הסתברותית מתוך אוסף ערכי פרמטרים), ובחרנו ברשימת ערכים זהה לזו של השאלה הקודמת.

לאחר בחירת 30 קבוצות אקראיות, קבוצת הפרמטרים שהביאו לדיוק הכי טוב (0.7251):

solver: adam

max_iter: 1000

hidden_layer_sizes: (128, 128)

alpha: 0.1

activation: tanh

ולכן נבחן פרמטרים אלו.

סיכום כיוונון שאלה 3 והרצת האלגוריתמים עם הפרמטרים הנבחרים על קבוצת המבחן:

Algorithm	Parameters chosen	Cross validation accuracy	Test accuracy
KNN	k_neigh 39	0.634	0.597
Decision Tree	max_depth: 9	0.74	0.746
SVM	C: 0.0464	0.708	0.710
SVM – poly kernel	C=10.0, degree =2	0. 713	0. 7188
SVM – rbf kernel	C=10, gamma=0.01.	0. 716	0. 7208
Random Forest	criterion: gini max_depth: 40 max_features: sqrt min_samples_leaf: 2 min_samples_split: 12 n_estimators: 350	0.745	0.752
AdaBoost	n_estimators: 350	0.754	0.759
Perceptron	penalty: l1 eta0: 0.063095	0.642	0.664
MLP	solver: adam max_iter: 1000 hidden_layer_sizes: (128, 128) alpha: 0.1 activation: tanh	0.7251	0.731

טבלה 3: סיכום כיוונון שאלה 3 והרצת האלגוריתמים עם הפרמטרים הנבחרים על קבוצת המבחן

דיון עבור טבלת הסיכום:

מטבלה זו ניתן לראות כי בחרנו פרמטרים טובים בתהליך כיוונון הפרמטרים, שכן כל הדיוקים על קבוצת המבחן הינם באותו סדר גודל של הדיוקים על קבוצת הוולידציה.

לדעתנו לעומת השאלה הקודמת, הדאטה שלנו עוד יותר לא פריד לינארית. הסקנו זאת מכך שאלגוריתמים המבוססים על הפרדה לינארי כמו SVM ו-Perceptron אינם מספקים דיוק טוב לעומת הדיוק של המסווגים הלא ליניאריים שבדקנו.

כמצופה אלגוריתם KNN הביא את הדיוק הנמוך ביותר כיוון שזהו האלגוריתם הכי בסיסי מכלל האלגוריתמים שבחנו.

בחרנו למערכת לסווג לפי AdaBoost כיוון שאלגוריתם זה הביא לדיוק מקסימלי עבור קבוצת הוולידציה ועבור קבוצת המבחן.

שאלה 4: חיזוי דירוג ELO של שחקנים לפי ביצועיהם

הכנת ה- DATA שאלה 4

את הדאטה שאספנו עבור שחקנים, חילקנו לרבעונים. כלומר ביצענו ממוצע על כל הנתונים של השחקן מרבעון נתון הקשורים לאיכות המשחק שלו. בנוסף הוספנו דאטה "יבש" על השחקן כגון גובה, גיל וכו' פר רבעון.

רשימת התכונות שאספנו עבור כל שחקן בכל רבעון בין השנים 2000-2019:

- elo_rank - דירוג ה-ELO של השחקן, המטרה שלנו
- bestRank - דירוג ה-ELO של השחקן הטוב ביותר עד לאותו הרבעון
- bestRankDate - תאריך בו התרחש דירוג השיא במדד ה-ELO של השחקן
- player_id - ת"ז של השחקן
- תכונות המעידות על איכות המשחק של השחקן באותו רבעון:
  df, svpt, bpSaved, bpFaced, SvGms, 2ndWon, 1stWon, 1stIn, ace
- Age - גיל השחקן
- Height - גובה השחקן
- loss_players_with_lower_atp_rank – כמות המשחקים בהן השחקן הפסיד לשחקנים בעלי דירוג ATP נמוך ממנו
- wins_players_with_lower_atp_rank - כמות המשחקים בהן השחקן ניצח שחקנים בעלי דירוג ATP נמוך ממנו.
- num_of_participent – מספר התחרויות בהן השתתף השחקן ברבעון הנוכחי
- num_of_participent_on_carpet - מספר התחרויות שהשחקן השתתף בהן אשר התקיימו על משטח סינטטי ברבעון הנוכחי
- num_of_participent_on_clay - מספר התחרויות שהשחקן השתתף בהן אשר התקיימו על משטח חימר ברבעון הנוכחי
- num_of_participent_on_grass - מספר התחרויות שהשחקן השתתף בהן אשר התקיימו על משטח דשא ברבעון הנוכחי
- num_of_participent_on_hard - מספר התחרויות שהשחקן השתתף בהן אשר התקיימו על משטח קשיח (בדר"כ אספלט) ברבעון הנוכחי
- num_of_wins – מספר הניצחונות ברבעון הנוכחי של השחקן
- num_of_wins_on_carpet - מס' הניצחונות של השחקן על משטח סינטטי ברבעון הנוכחי
- num_of_wins_on_clay - מס' הניצחונות של השחקן על משטח חימר ברבעון הנוכחי
- num_of_wins_on_grass - מס' הניצחונות של השחקן על משטח דשא ברבעון הנוכחי
- num_of_wins_on_hard - מס' הניצחונות של השחקן על משטח קשיח (בדר"כ אספלט) ברבעון הנוכחי
- q – באיזה רבעון אנחנו בשנה.
- year- שנה הנוכחית.

ניתוח הדאטה

כיוון שרוב האלגוריתמים משתמשים ברגולריזציה נרצה שהטווחים של התכונות יהיו זהים ולכן ננרמל אותם לפי נרמול mini-max.

מתודולוגיה ניסויית שאלה 4 – פתרון בעזרת רגרסיה

עבור השאלה הרביעית, נרצה למצוא את אלגוריתם ה-Regression האופטימלי, לכן נכוונן את הפרמטרים שציינו לכל אלגוריתם.

נשתמש ברבעונים אחרונים של מס' שנים שישמשו כקבוצות ולידציה.

בכל פעם, ניקח רבעון רביעי של שנה אחת שתהיה קבוצת הוולידציה שלנו, נלמד את כלל הרבעונים הקודמים בעזרת המודל שאנו בוחנות.

כיוון שלא ניתן להשתמש בביצועים של שחקנים ברבעון הנוכחי בתור סט ולידציה (שכן דבר זה שקול ללדעת את העתיד), ניצור סט חדש. לכל שחקן ניצור ממוצע של הביצועים שלו בארבעת הרבעונים הקודמים לרבעון הנוכחי ונשתמש במידע כסט הוולידציה.

נדגיש כי בנייה זו מסתמכת על כך שבדרך כלל ביצועים של שחקנים לא משתנים הרבה מרבעון לרבעון.

נשווה בין הדירוג שקיבלנו לדירוג האמיתי של השחקן בסוף הרבעון ונבצע שקלול MSE ו-R^2 על מנת להבין מה טיב המודל עם הפרמטרים שבחנו.

על מנת לא להסתמך על רבעון אחד כשנבחר פרמטרים, נעשה את השקלול עבור מס' שנים (מציאת MSE, R^2) ונחשב את השקלול הממוצע.

בנוסף נחשב את הMSE ו-R^2 של מודל Dummy (מודל שמחזיר את הדירוג הממוצע בכל פעם) על מנת לקבל אינדיקציה לטיב התוצאות שנקבל.

MSE (mean square error) – ממוצע ריבועי של הטעויות, כלומר ההבדל בין הפלט של המודל לערך האמיתי שרצינו לחזות.

R^2– קורלציה בין מה שחזינו לערך האמיתי. ככל שהוא יותר קרוב ל-1 כך המודל חוזה יותר טוב.

לפני ביצוע הניסויים חישבנו את תוצאת מודל הDummy וקיבלנו את התוצאות הנ"ל:

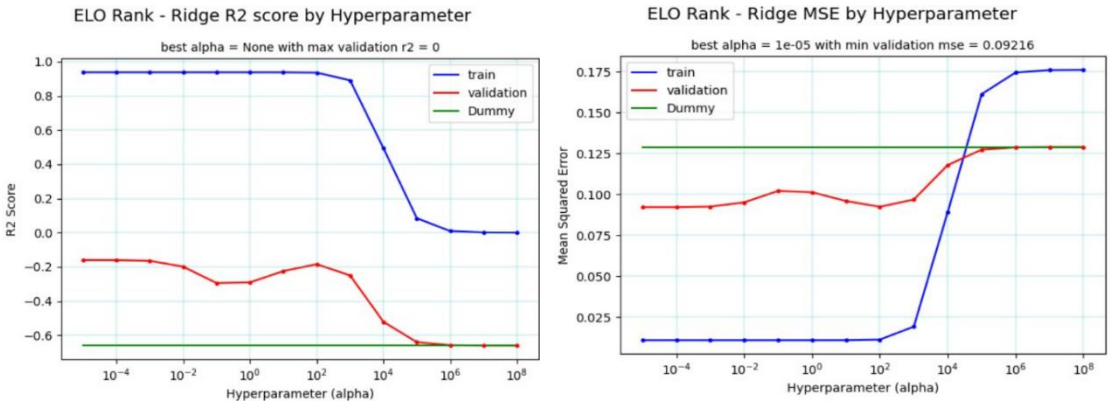
average_dummy_mse = 0.12891973410337926

average_dummy_r2 = -0.6610280962482579

כיוונון פרמטרים

<u>Ridge Regression:</u>

Validation Best parameters: λ=10^{-5} with MSE: 0.092

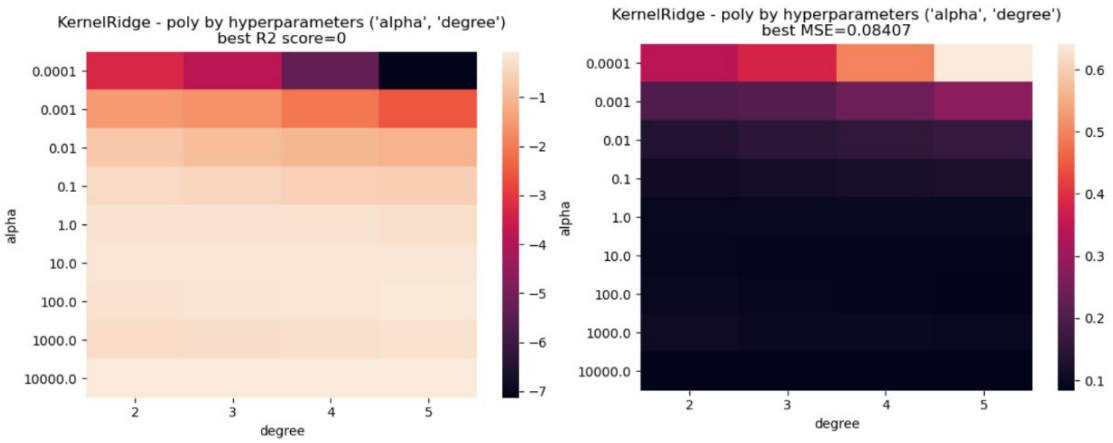


גרף 32: כיוונון Ridge Regression-שאלה 4

אפשר לראות מהגרף כי המודל שלמדנו יותר טוב ממודל ה-Dummy עבור שני הקריטריונים שבחנו. בשני הגרפים עבור $\lambda = 10^{-5}$ התוצאה הייתה מיטבית. עם זאת נראה כי בקריטריון R^2 לא עברנו את 0 (אפילו לא קרובים ל-1) דבר שמעיד על ריחוק מפתרון לבעיה.

:Ridge Regression -poly kernel

Validation Best parameters: λ=10^3, Degree =2 with MSE: 0.084



גרף 33: כיוונון Ridge Regression- poly kernel -שאלה 4

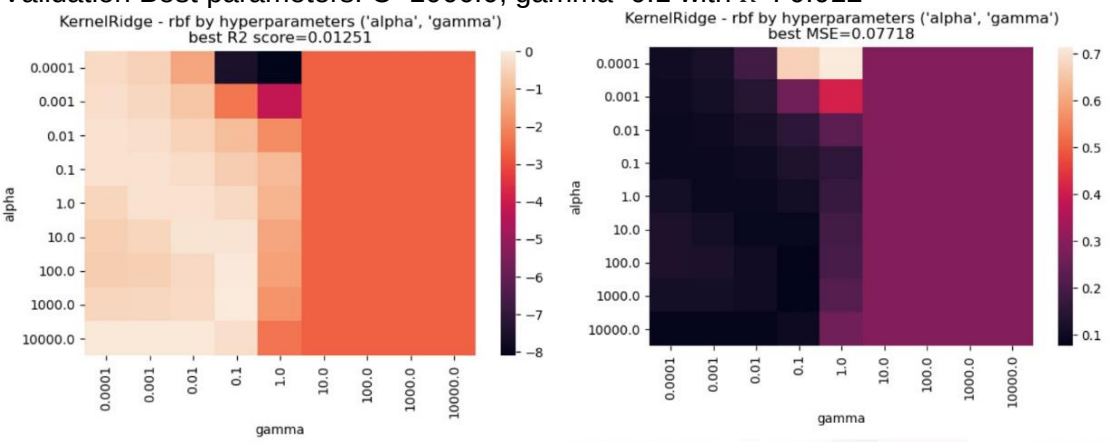
אפשר לראות מהגרף כי המודל שלמדנו יותר טוב ממודל ה-Dummy עבור שני הקריטריונים שבחנו. ככל ש λ יותר גדול כך יש עליה בדיוק.

ב-MSE עבור $\lambda = 10^3$ ו Degree =2 קיבלנו את התוצאה המיטבית. עם זאת נראה כי בקריטריון R^2 לא עברנו את 0 (אפילו לא קרובים ל1) דבר שמעיד על ריחוק מפתרון לבעיה.

:Ridge Regression - rbf kernel

Validation Best parameters: C=1000.0, gamma=0.1 with mse: 0.077

Validation Best parameters: C=1000.0, gamma=0.1 with R^2: 0.012



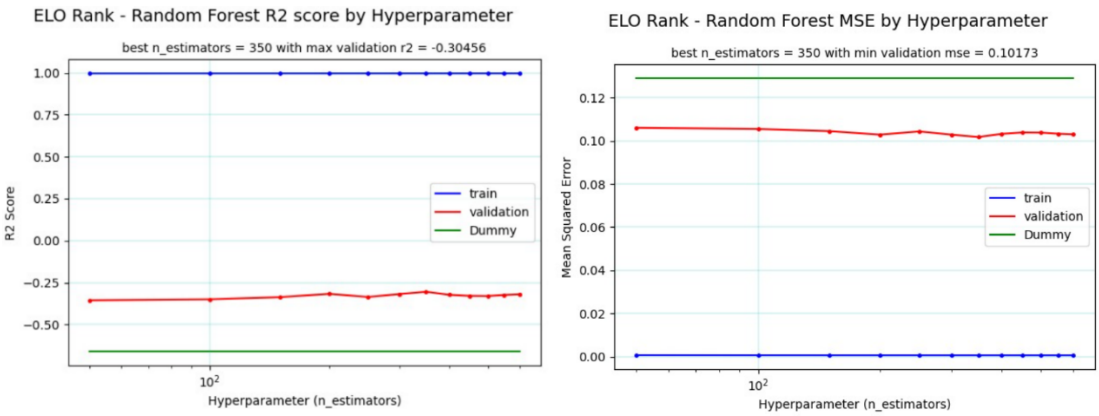
גרף 34: כיוונון Ridge Regression- rbf kernel -שאלה 4

גם כאן המודל שלמדנו יותר טוב ממודל ה-Dummy עבור שני הקריטריונים שבחנו. כאשר gamma קטן ניתן לראות כי התוצאות יותר טובות.

בשני הקריטריונים הפרמטרים הכי טובים הינם זהים. בנוסף עברנו את הרף ה-0 בR^2.

:Random Forest

Validation Best parameters: n_estimators=350 with mse: 0.101
Validation Best parameters: n_estimators=350, with R^2 : -0.304

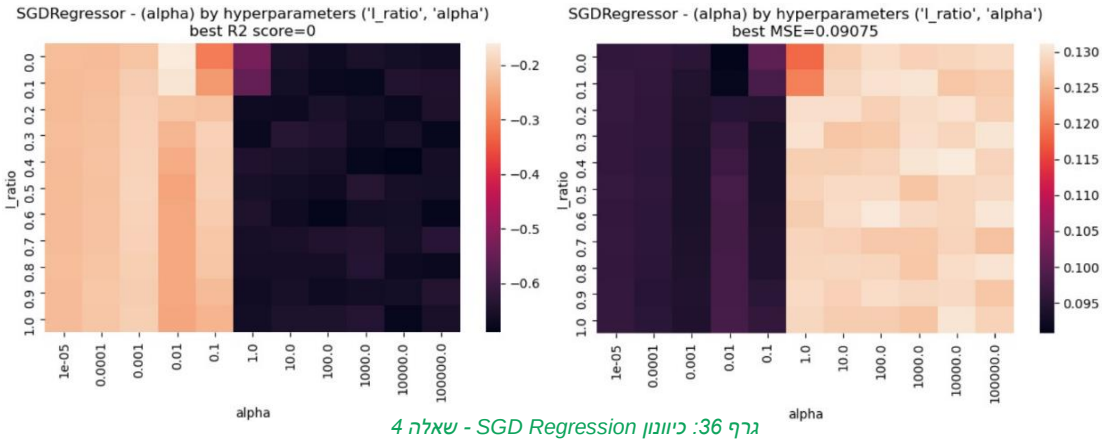


גרף 35: כיוונון Random Forest - שאלה 4

בשני הקריטריונים הפרמטרים הכי טובים הינם זהים, אך לא עברנו את הרף ה-0 בR^2.
עבור כל ערכי הפרמטרים שבדקנו התוצאות יחסית זהות, ניתן לראות זאת מהגרף היציב.

:SGD Regression

Validation Best parameters: l1_ratio=0, alpha=0.01 with mse: 0.090



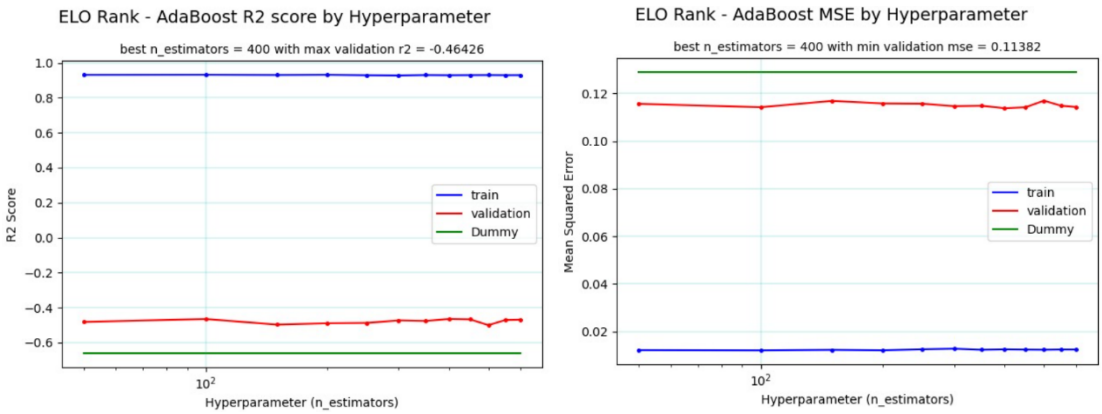
גרף 36: כיוונון SGD Regression - שאלה 4

התוצאה הכי טובה שקיבלנו הינה עבור l1_ratio=0,כלומר עבור רגולריזציה L2 בלבד. עבור alpha קטנות התוצאות טובות יותר, אך עדיין לא עברנו את הרף ה-0 בR^2.

:AdaBoost

Validation Best parameters: n_estimators = 400 with mse: 0.113

Validation Best parameters: n_estimators = 400 with R^2: -0.462



גרף 37: כיוונון AdaBoost - שאלה 4

בשני הקריטריונים הפרמטרים הכי טובים הינם זהים, אך לא עברנו את רף ה-0 בR^2.

עבור כל ערכי הפרמטרים שבדקנו התוצאות יחסית זהות, ניתן לראות זאת מהגרף היציב. בנוסף התוצאות על סט הוולידציה קרוב ל-Dummy שזה אומר שאין לאלגוריתם זה כמעט שום יתרון על פתרון "טיפשי".

סיכום כיוונון שאלה 4:

Algorithm	Parameters chosen	Validation mse	Validation R^2
Dummy	-	0.128	-0.661
Ridge Regression	λ=10^{-5}	0.092	-0.160
Ridge Regression -poly kernel	λ=10^{3}	0.084	-0.067
Ridge Regression - rbf kernel	C=1000.0, gamma=0.1	0.077	0.012
Random Forest	n_estimators=350	0.101	-0.304
SGD Regression	l1_ratio=0, alpha=0.01	0.090	-0.157
AdaBoost	n_estimators=400	0.113	-0.462

טבלה 4: סיכום כיוונון שאלה 4

המודל הטוב ביותר שקיבלנו הוא Ridge Regression - rbf kernel מבחינת שני הקריטריונים. אנו לא מרוצות מהתוצאות כיוון שניקוד ה-R^2 רחוק מאוד מ1.

לכן, נפנה לדרך אחרת לחישוב הדירוג. נשתמש במודל הטוב ביותר שקיבלנו בשאלה 3 על מנת לחזות את תוצאות המשחקים ברבעון מסוים ומכך לגזור את הדירוג שיתקבל בסוף הרבעון.

נספר בקצרה איך ELO עובד: שני שחקנים נכנסים לתחרות עם דירוג שמבוסס על תוצאותיהם הקודמות. ELO משתמש בדירוג המוקדם הזה כדי לחזות את תוצאת מפגש הראש בראש שלהם, ובכך "להכתיר" פייבוריט טרם ההתמודדות. לאחר מכן, ובהתחשב בתוצאה, הוא משתמש בתחזית המוקדמת כדי לעדכן את הדירוג שלהם. המשמעות בפועל היא כזו – ניצחת משחק שהיית "אמור" לנצח? תרוויח נקודות ותתקדם בדירוג אבל לא יותר מדי. הפתעת שחקן שעדיף עליך משמעותית? תקבל הרבה יותר נקודות ותזנק בדירוג. כמות הנקודות שתרוויח היא זו שתילקח מהמפסיד, וככזו – תהיה תלויה תמיד במצב בו יריבך ואתה נכנסתם לתחרות.

הנוסחאות עבור חישוב דירוג ELO שנשתמש בהם לחיזוי שלנו, לאחר שקראנו במאמרים[1] שונים:

- הסתברות של שחקן לנצח בהינתן ניקוד דירוג ה-ELO שלו ושל השחקן מולו הוא משחק:

$$p_{\text{win}} = \frac{1}{1 + 10^{\frac{elo_{\text{opponent}} - elo_{\text{player}}}{400}}}$$

- חישוב ניקוד הדירוג החדש של כל שחקן בסוף משחק:

$$\text{elo(new)} = \text{elo(last)} + \text{K(outcome} - p_{\text{win}})$$

outcome - אינדיקטור של תוצאת המשחק. ערכו 1 אם השחקן ניצח ו-0 אם הפסיד.

K - פרמטר אשר מצביע על כמה דירוג ה-ELO צריך להשתנות בעקבות תוצאת המשחק.

מתודולוגיה ניסויית שאלה 4 - פתרון על סמך חיזוי תוצאות ואלגוריתם לדירוג

ניקח את הדאטה על משחקי טניס שהתרחשו, אותם אספנו בשאלה 3. כל רבעון החל מתחילת 2005 ישמש כמבחן וכלל הרבעונים הקודמים לו ישמשו לאימון.

נרצה לחזות את כלל התוצאות של המשחקים ברבעון הנבחר ולפיהן לחשב את דירוג ה-ELO של כלל השחקנים. מכיוון שחלק מהחיזוי של תוצאות משחק צריך את תכונות הדירוגים של השחקנים בתחילת המשחק, נשנה את הפרמטרים להיות הדירוגים האחרונים שלהם לפני תחילת הרבעון כדי למנוע "ידיעת העתיד".

לאחר שהכנו את הדאטה, נחזה את תוצאות המשחקים ברבעון לפי מודל AdaBoost שמצאנו כטוב ביותר בשאלה 3.

נעבור בצורה כרונולוגית על המשחקים ברבעון ובאמצעות התוצאה שקיבלנו מהחיזוי נחשב את הדירוג (בעזרת הנוסחאות שציינו למעלה). לאחר קבלת הדירוגים הסופיים נשווה אותם לדירוגים האמיתיים בסוף הרבעון עבור כל שחקן.

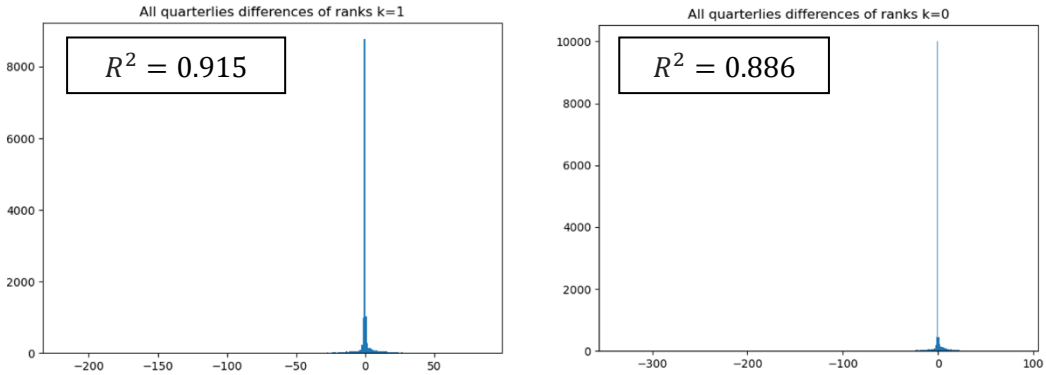
ניתן לראות מהנוסחאות שכתבנו מעלה כי K הינו משתנה שאנחנו צריכות לקבוע. לאחר קריאת מאמרים[2] באינטרנט ראינו כי הרבה מהם מצביעים על כך ש K=32 הינו ה-K האופטימלי אך למרות זאת נבחן את ה-Kים הבאים:

$$K = [0, 1, 2, 4, 6, 8, 10, 16, 32, 64, 128]$$

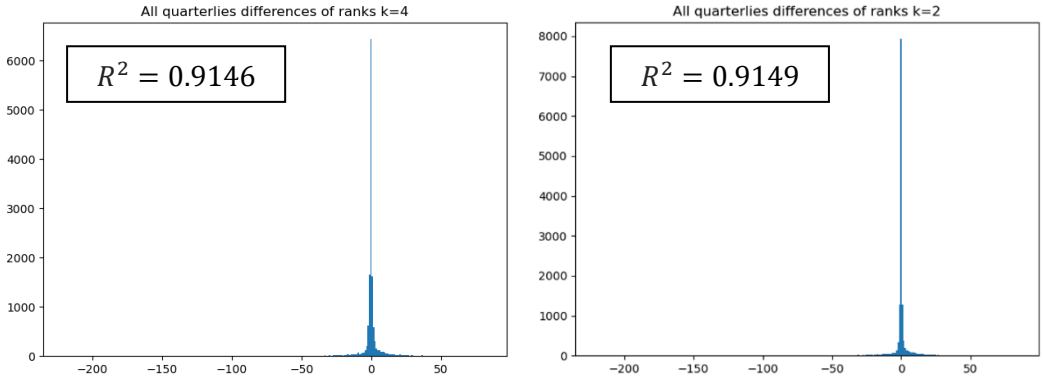
עבור כל ניסוי נציג את התפלגות הטעויות בדירוג (ההפרש מהדירוג האמיתי) עבור כל הרבעונים. קיבלנו את הגרפים:

[1] http://www.tennisabstract.com/blog/2019/12/03/an-introduction-to-tennis-elo/
https://www.betfair.com.au/hub/tennis-elo-modelling/

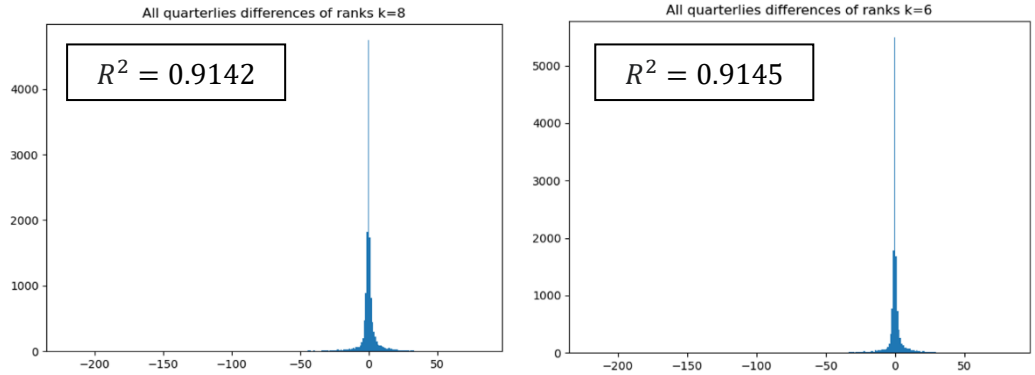
[2] https://www.ultimatetennisstatistics.com/blog?post=eloKfactorTweaks



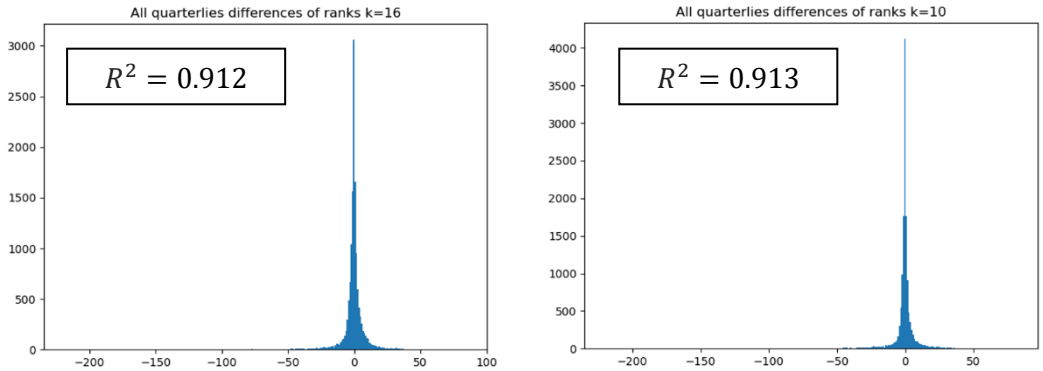
גרף 41: הפרש הדירוגים בין החיזוי לאמת עבור k=0 ו k=1



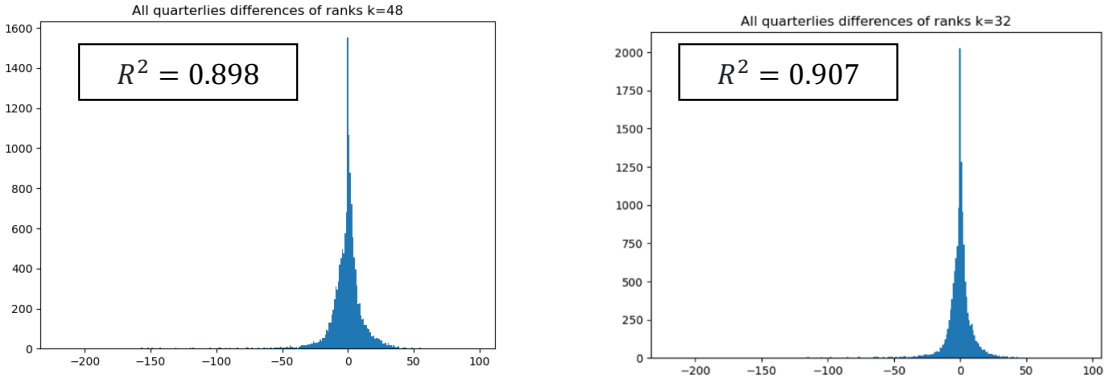
גרף 40: הפרש הדירוגים בין החיזוי לאמת עבור k=2 ו k=4



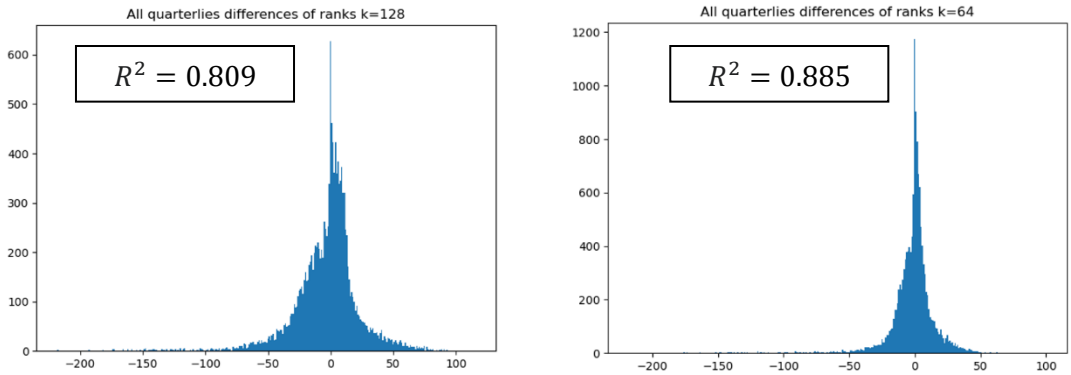
גרף 39:הפרש הדירוגים בין החיזוי לאמת עבור k=6 ו k=8



גרף 38:הפרש הדירוגים בין החיזוי לאמת עבור k=10 ו k=16



גרף 43: הפרש הדירוגים בין החיזוי לאמת עבור k=32 ו k=48



גרף 42: הפרש הדירוגים בין החיזוי לאמת עבור k=64 ו k=128

ניתן לראות כי עבור K=0 קיבלנו את כמות הדיוקים (הפרש 0 בין הניבוי לדירוג האמיתי) הגבוהה ביותר. כלומר ניתן להסיק כי דירוגים לא נוטים להשתנות מרבעון לרבעון שאחריו. אבל עבור K=1 קיבלנו את ניקוד R^2 הגבוה ביותר. לכן בכל זאת יש תועלת באלגוריתם שפיתחנו ואנו משערות שעבור Kים גדולים יותר קיבלנו תוצאות טיפה פחות טובות מכיוון שחישוב הדירוג משתמש בחיזוי עפ"י שאלה 3 (שהינו בדיוק של פחות מ80%) מה שגורם לכך שחלק מתוצאות המשחקים שגוייות.

דיון ומסקנות על שאלות 1-4

למדנו מפרויקט זה בעזרת שלושת השאלות הראשונות האם האלגוריתמים אותם בחנו יודעים להבדיל, לתת חשיבות גדולה, לתכונות אשר נראות לנו הגיוניות, כמו תוצאת כל משחקון, אשר מצביעות לנו על תוצאת המשחק באופן ישיר. ואולי להצביע לנו על תכונות שכן משפיעות על חיזוי תוצאת משחק טניס אך לא נראות לנו טריוויאליות.

בשאלה 1 למרות שציפינו כי הדיוק עבור רוב האלגוריתמים יהיה 1 (זאת מכיוון שיש לאלגוריתמים את תוצאות המשחק), קיבלנו כי הדיוק המרבי הינו קרוב מאוד ל1 אך לא 1 בדיוק. מספר סיבות שאנו משערות שיכולות לגרום לדיוק הנ"ל:

- האלגוריתמים לא מספיק "חכמים"
- שגיאה בשלב ההכנה של הדאטה: בחירת לא נכונה של צורת הנרמול/ מילוי ערכים חסרים, הוספת תכונות שלא מעילות או גורמות להטעיה.
- טעות בדאטה שאספנו.

ברוב האלגוריתמים ראינו כי התכונות אשר נבחרו כמשמעותיות ביותר היו אלה שציפנו, מה שמצביע על כך שהאלגוריתמים לא טיפשים.

בשאלה 2 ראינו כי באמת האלגוריתמים שלנו נותנים חשיבות גדולה לאיכות המשחק של שחקן במהלך המשחק, מה שמצביע בעקיפין על התוצאה, כמו הACEים וכו'.

בשאלה 3 לא ידענו למה לצפות וזאת מכיוון שאם היה אלגוריתם טוב לניבוי הריי שהיו מפסיקים עם הימורים בטניס כי ניבוי התוצאות היה מספיק טוב. כצפוי התכונה המשמעותית ביותר הייתה הדירוג של השחקנים המתחרים אך תכונה זו לא מספיק טובה על מנת לנבא בדיוק שגבוה מ-80%.

בשאלה 4 ציפינו כי רגרסיה תיתן לנו תוצאות טובות לניבוי אך כיוון שמדד R^2 הראה כי התוצאה לא קורלטיבית אנו מניחות כי הניבויים בעזרת רגרסיה לא מספיק טובים ולכן פנינו לדרך השנייה. בדרך השנייה לפתירת שאלה 4 ראינו כי הערך של K אותו קראנו במאמרים אינו האופטימלי עבורנו כיוון ששימוש בשאלה 3 יוצר ניבויים שגויים שמשפיעים מאוד עבור ערכי K גדולים.

כיוונים להמשך המחקר

היינו רוצות להמשיך ולחקור על עוד אלגוריתמי למידה שאולי יתנו לנו תוצאות יותר טובות, להעמיק ולפצח את ניבוי משחקי הטניס רק עם תכונות יבשות שיש לנו בתחילת משחק, שכן זה המידע היחיד שאנו יודעים בוודאות בתחילת משחק טניס.

כיוון נוסף, להסתכל על תכונות נוספות שלא אספנו כאן כגון פציעות של שחקנים, כמה זמן עבר מאז הפציעות, תקופת חופשה, כמות ילדים, מצב בחיים (נשוי, רווק וכדומה),מאיזה גיל התחיל להתאמן, מאיפה המימון שלו, כמה מממנים אותו, מאמן כישורי טניס, מאמן כושר, פיזיותרפיסט, פסיכולוג מקוצעי, סוג מחבט וכו'.

בנוסף היינו שמחות בעיקר כנשים ליצור מודלים גם עבור שחקניות טניס. לראות אולי האלגוריתמים שבחנו כן טובים בניבוי בקהילה הנשית. בפרויקט זה לא בנינו מודל עבור נשים מפאת קוצר זמן חישובי שהיה מוכפל ב-2.

את הפרוייקט שלנו ניתן למצוא ב-GitHub בכתובת:

https://github.com/shacharkag/ai_project

ביבליוגרפיה

מקורות נתונים:

- https://github.com/JeffSackmann.
- http://www.tennis-data.co.uk/alldata.php
- https://www.ultimatetennisstatistics.com/eloRatings

טניס:

- https://he.wikipedia.org/wiki/%D7%98%D7%A0%D7%99%D7%A1
- https://www.tennis.org.il/%D7%9E%D7%99%D7%9C%D7%95%D7%9F-%D7%9E%D7%95%D7%A0%D7%97%D7%99-%D7%98%D7%A0%D7%99%D7%A1/

דירוג שחקני טניס:

- https://tennis360.co.il/%D7%93%D7%99%D7%A8%D7%95%D7%92-%D7%94%D7%98%D7%A0%D7%99%D7%A1-%D7%94%D7%A2%D7%95%D7%9C%D7%9E%D7%99-%D7%9B%D7%9C-%D7%9E%D7%94-%D7%A9%D7%A8%D7%A6%D7%99%D7%AA%D7%9D-%D7%9C%D7%93%D7%A2%D7%AA/
- https://tennis360.co.il/%d7%9e%d7%93%d7%95%d7%99%d7%a7-%d7%9e%d7%94-atp-%d7%94%d7%90%d7%9d-%d7%96%d7%94%d7%95-%d7%93%d7%99%d7%a8%d7%95%d7%92-%d7%94%d7%98%d7%a0%d7%99%d7%a1-%d7%94%d7%90%d7%9e%d7%99%d7%aa%d7%99/

מדד ה-ELO:

- http://www.tennisabstract.com/blog/2019/12/03/an-introduction-to-tennis-elo/
- https://www.betfair.com.au/hub/tennis-elo-modelling/
- https://www.ultimatetennisstatistics.com/blog?post=eloKfactorTweaks

מפתח גרפים, איורים וטבלאות

גרפים

רשימת איורים

רשימת טבלאות