חיזוי תוצאות משחקי כדורגל

פרויקט בבינה מלאכותית

236502

הפקולטה למדעי המחשב, טכניון

מגיש : מחמוד נסאר

ת.ז : 318489200

מייל : mahmoud-nas@campus.technion.ac.il

מבוא :

משחק כדורגל הוא משחק של 90 דקות, משתתפים בו שתי קבוצות שכל אחת מהן מורכבת מאחד עשר שחקנים, כדי שקבוצה כלשהיא תנצח היא צריכה להפקיע יותר שערים מאשר השנייה.

כדורגל הוא ענף הספורט הכי נפוץ על פני כדור הארץ, כ- 240 מיליון איש ברחבי העולם משחקים כדורגל באופן סדיר, בין אם באופן מקצועני ובין אם חובבני. במסגרת הענף פועלים כ-300,000 מועדונים מקצועיים, הרשומים בגופים הרשמיים המנהלים אותו.

בשנים האחרונות עם התקדמות הטכנולוגיה היא התחילה להופיע באופן מובהק בתוך ענף הכדורגל, קבוצות כדורגל, שופטים, וגם התאחדויות כדורגל התחילו להיעזר באמצעים טכנולוגים לטובת אנליזה, קבלת סטסטיקה, וגם בעבודה הישירה שלהם. למשל בשנת 2018 התווספה טכנולוגית ה- VAR (video assistant refree), שהיא בעצם שופט וידאו, הסוקר החלטות שנעשו על ידי השופט הראשי בזמן אמת, תוך שימוש בקטעי וידאו ובמצלמות מזוויות שונות.

בנוסף לכל קבוצה כדורגל יש קבוצה של אנלסטים שמשתמשים בטכנולוגיה ובמידע האישי שלהם,כדי להכין למשחק הבא.הם עושים אנליזה של משחקי הקבוצה שלהם וגם של משחקי הקבוצה הנגידת, כדי לדעת את מעמדם לקראת המשחק, איזה נקודה חולשה יש להם ולקבוצה הנגדית, **מה התוצאה הצפויה למשחק,** ואם הצפי שהם יפסידו, מה אפשר לעשות כדי לשנות את זה. למשל מאמנים חכמים כשהם יודעים שהסיכויים שלהם לניצחון חלשים, הם משנים את תכנית המשחק שלהם, לפעמים הם עוברים לתכנית משחק הגנתי מאודומנסים להפתיע התקפות מתפרצות.

חיזוי של תוצאת משחק היא בעיה קשה מאז ומתמיד, אם נגיד למשל שיש משחק מול קבוצה גדולה כמו ברצלונה וקבוצה קטנה כמו אלצ"י, במבט ראשון אולי מצפים שברצלונה תנצח, אבל אם היתה ברצלונה באה אל המשחק בתקופה קשה עם הרבה הפסדים רצופים, ואלצ"י להפך, מי הייתם מציפם שתנצח ?

בפרויקט שלי ניסתי ליצור כלי למעקב אחרי קבוצות וחיזוי של תוצאות המשחקים (ניצחון/הפסד), לצורך מטרה זו בחנתי מספר אלגורתמי *Machine Learning* שונים. והשתמשתי בהרבה פקטורים כמו היסטורית הקבוצות, מעמדם בליגה וכו"...

פרויקט כזה יכול להווה כלי עזר עצום לצוות הטכני והמנהלי של קבוצות כדורגל, נדמיין מצב שבו המאמן בתחילת ההכנה של המשחק כבר יודע (באחוז וודאות גבוה) את התוצאה של המשחק, זה יכול לשנות את הדרך בו הוא מסתכל על

המשחק, ואת דרך ההכנה למשחק, הרכב השחקנים הפותח ועוד הרבה דברים, ואולי בזה הוא אכן יכול לנצח משחק שהחיזוי שלו היה הפסד.

<u>תיאור הפתרון המוצע לבעיה :</u>

אני מנסה לפתור בעיית חיזוי, ולכן החלטתי להשתמש באלגותמי למידה כדי לפתור את הבעיה הנ"ל, החלטתי לבחון את התוצאות של מספר אלגורתמים שונים ולא רק אחד, כי לשימוש בכל אלגוריתם שונה יש יתרונות וחסרונות שונים.

בתהליך פתרון בעיות למידה יש מספר שלבים :

- **איסוף מידע:** אני יכול להגיד שתהליך זה היה התהליך הכי מאתגר בפרוייקט, למרות שכדורגל הוא ענף ספורט מאוד פופיולארי ולכן מצפים שיש הרבה מקורות לאסוף מידע, אבל בפועל בשביל פרויקט זה אנחנו צריכים מידע מדויק ומורחב על משחקים שעבר על קיומן מספר שנים, ולכן אין הרבה אתרי אינטרנט שמספקים את זה. מקורות איסוף המידע :
  - *Flash Score* : מאתר זה נלקח את רוב המידע כדי לבנות את הדוגמאות. אני באיסוף המידע התרכזתי במשחקי הליגה הספרדית (למרות שהפרוייקט יכול לפעול במידה שווה על כל ליגה אחרת), התרכתזי בשתי עונות של הליגה הספרדית, עונת 2017/2018 וגם 2018/2019. איסוף המידע נעשה בעזרת תוסף *google chrome* בשם *web scrapper*, עבור כל עונה תוצאת איסוף המידע היתה קובץ בשם *games.csv* שכל שורה בו מכילה פרטים על משחק שהתקיים בעונה, עבור כל משחק נלקח המידע הבא : **תאריך ושעת המשחק, שם הקבוצה הראשונה (קבוצה ביתית), שם הקבוצה השנייה (קבוצת חוץ), תוצאת המשחק, כמה קהל היה במשחק, שם האסטדיון בו קרה המשחק, ובאיזה סיבוב התרחש המשחק (הליגה הבפרדית יש 38 סיבובים) .**
  - *Transfer Market*: מאתר זה נלקח מידע על **השווי של הקבוצות בשוק** בתחילת כל עונה, מידע זה היה יותר כל לחלץ ידנית, מאחר ובכל עונה יש 20 קבוצה, כלומר יש בסה"כ 40 ערך לחלץ. הערכות מהסוג הזה נעשות בתחילת כל עונה, לכן עבור עונה קבוצה כלשהי ועונה כלשהי יש ערך אחד ויחיד בשוק. מידע זה נשמר (עבור כל עונה) בתוך קובץ בשם *teams.csv* שמכיל את כל המידע שאצטרך עבור הקבוצות כולל מידע זה.
  - *FIFA Index*: מאתר זה נלקח את המידע **על דירוג הקבוצות (שנוצר בעזרת טיב שחקניה)**, ה- *FIFA* בתזמון עם תחילת כל עונה מציגה ניקוד של שחקן (מתוך 100) שהוא איזושהי הערכה של כמה השחקן טוב, הניקוד הזה נקבע על ידי בדיקה של ה *FIFA* ליכולת השחקן בשנה האחרונה, בדיקתם מסתמכת על הרבה פקטורים, בינם

: יכולת הבעיטה, קצב ,מהירות, שימוש בראש, וכו". ועל כך - ובעזרת ניקוד השחקנים - יש להם גם הערכה ברמת הקבוצות, כלומר יש להם ניקוד לכל קבוצה, שמהווה מדד לכמה הקבוצה טובה וחזקה. גם פה המידע נאסף באופן ידני בגלל שהוא לא כל כך גדול.

- wikipedia : בעזרת אתר זה נעשה השלמה של המידע הסר עבור הקבוצות עצמן, עבור כל קבוצה התווספו התכונות הבאות : **האסטדיון שלה**, **מספר תארי הליגה הספרדית** שיש לה, **מספר תארי הליגה הארובית** שיש לה (europa leage) **, מספר תארי ליגת הצ"מפיונס** שיש לה (champinos leage).

- **בחירת התכונות**: בחירת התכונות היא משהו מאוד חשוב שמשפיע באופן ישיר על התוצאה הסופית, והם נבחרו כאשר נלקח בחשבון מספר נקודות מרכזיות שמביניהם :

  - היה צורך בבחירת התכונות הכי משפיעות על תוצאות המשחקים. באותו זמן אי אפשר לקבל מידע על כל תכונה שרוצים, ולכן יש פה איזשהו *tradeoff* שצריך לאזן באופן שמניב הכי תעולת תוך שימוש בתכונות אפשריות להשגה.

  - צריך שיהיה תכונות שמשקפות את אופי הקבוצה, כלומר האם היא מבין הקבוצות הגדולות שמצופה ממנה לנצח ברוב המשחקים שלה. וצריך גם שיהיו תכונות שמשקפות את ההיסטוריה הקרובה של הקבוצה, כלומר שמשקפות את המומנטום הנוכחי שיש לקבוצה.
  - חוץ מלהסתכל על ההישגים של הקבוצה, צריך גם להסתכל על סגל השחקנים שיש לה העונה, כי למשל אולי ברצלונה היא אחד המועדונים הגדולים בספרד ובאירופה אבל אם סגל השחקנים שלה העונה חלש, אזי מצופה ממנה יכולת יותר חלשה מהרגיל.

עבור כל דוגמא (משחק) **יש בסך הכך 16 תכונה** :

נזכור ש *team*1 הוא הקבוצה המארחת, ו- *team*2 היא הקבוצה האורחת.

1- team1 history points - ניקוד שמשקף את תוצאות חמשת המשחקים האחרונים של קבוצה 1.
2- team1 market value - השווי בשוק עבור קבוצה 1.
3- Audience – מספר הקהל שנמצא ביציעים.
4- team1 table position – מיקום קבוצה 1 בטבלת הליגה לקראת המשחק.
5- team1 league titles – מספר תארי הליגה הספרדית שיש לקבוצה 1.

6- team1 champions league titles - מספר תארי ליגת הצ"מפיונס שיש לקבוצה 1 (champinos leage).
7- team1 europa league titles – מספר תארי הליגה הארובית שיש לקבוצה 1 (europa leage).
8- team1 Rank – הדירוג של קבוצה 1, זאת אומרת ממוצע הדירוג שיש לסגל השחקנים העונה.

ועבור קבוצה 2 יש גם אותם 8 תכונות.

- **קלסיפיקציה או רגרסיה :** היה אפשר להסתכל על הבעיה כבעיית רגרסיה, כלומר ביהנתן משחק לדעת לחזות תוצאה ספציפית, אם קבוצה כלשהיא תנצח אז בכמה היא תנצח ומה תהיה התוצאה הסופית.
החלטתי להסתכל על הבעית כבעיית סיווג בינארי, כלומר בהינתן משחק צריך לסווג אותו לערך מבין השניים : 1. הקבוצה המארחת תנצח 2. הקבוצה האורחת תנצח, כלומר אפילו תיקו לא נלקח בחשבון בזמן סיווג הבעיה. הסיבה לדבר כזה היא שגם עם הצמצום, הבעיה מאוד מורכבת לפתרון, בסופו של דבר זה משחק כדורגל שהתוצאה הסופית שלו יכולה להשתנות בשנייה, פריטם קטנים יכולים להפוך את התוצאה הסופית (כמו קבלת כרטיס אדום \ חלוץ לא מרוכז\ טעות שוער...), והכנסת תוצאת תיקו כתוצאה אפשרית היה רק מסבך את העיניים.
- **עיבוד הנתונים :** רוב העבודה בשלב זה הייתה בשביל יצירת המידע שלא נמצא במפורש בנתונים שנאספו מהאתרים. היה צורך ביצירת מידע שמעיד על מצב כל קבוצה לקראת המשחק, למשל מיקום הקבוצה בטבלה, מידע זה לא נכלל במידע שנאסף, ואין אף אתר אינטרנט מספק מידע על מיקום הקבוצות בטבלה **לקראת המשחק**, יש רק תיאור הטבלה בסוף הליגה.
לכן כדי ליצור מידע כזה היה צורך לעשות הרצה שמסמלצת את העונה כולה, ואז לשמור את המידע בכל נקודת זמן רצויה. כלומר לעשות מערכת שקוראת את המשחקים משחק אחרי השני (מתוך הרובץ *games.csv*), ואחרי קריאת כל משחק, היא מיישמת את השפעת תוצואת המשחק על המערכת (למשל שינוי הסדר בטבלת הליגה), ואז לקראת קריאת המשחק השני, יש במערכת עצמת את כל המידע שצריך, שמשקף את מצב הקבוצות בדיוק לפני תחילת המשחק. וכך אפשר לשלוף מהמערכת את כל המידע שצריך עבור 16 התכונות של המשחק וליצור דוגמא שלמה עבור המשחק הזה.
בזמן התחלת המימולציה ובזמן שהמערכת עדיין לא קיבלה אף משחק, יש חלק סטטי של מידע שמוכנס לתוך המערכת, למשל עבור כל קבוצה נוצר אוביקט *Team* שמכיל בתוכו את כל המידע הסטטי (שלא משתנה לאורך כל העונה) של הקבוצה מידע זה נלקח מקובץ *Team.csv*.

המערכת שמסמלצת את העונה היא בעצם שני *classes*, הראשון בשם *Team* שישמור בתוכו את כל המידע שצריך עבור קבוצה כלשהיא בליגה. והשני הוא *League* שמתאר עונה שלמה, מכיל בין היתר מבנה נתונים של *Team*. הפונקציה processData היא הפונקציה שמריצה את הסומלציה שבתורה עושה עיבוד לנתונים ומייצרת קובץ בשם processedGames.csv שמכיל את הדוגומאות בתוך התקייה dataset.

בתוך המערכת הנ"ל יש טיפול במספר תכונות בפני עצמן בהתחשובה בנרמול והעברת המידע לערך מספרי, באופן הבא:

- עבור התכונה team1 history points : לקראת כל משחק, כפי שהסברתי קודם, אפשר לגשת למידע שיש במערכת על הקבוצה הזו, ולחלץ את תוצאות חמשת המשחקים האחרונים של הקבוצה, הקבוצה תקבל את הניקוד לפי המפה הבאה :

משחק	ניקוד להפסד	ניקוד לתיקו	ניקוד להפסד
אחרון	4	2	-3
לפני אחרון	3	1	-2
לפני 3 משחקים	3	1	-2
לפני 4 משחקים	2	0	-1
לפני 5 משחקים	1	0	-1

ערך התכונה הזו של הקבוצה יהיה סכום הניקוד שינתן לה עבור כל משחק.

- עבור התכונה team1 market value נעשה נרמול בגלל לתכונה זו יש ערך הרבה גדול מהתכונות שונות (כדי למנוע מצב שבו התכונה הזו משתלטת על תוצאת המסווג). ערך זה בדרך כלל מתואר בעשרות מילונים, ולכן הוחלט לחלק אותו במליון.
- Audience – הערך של תכונה זו הוא בעשרות אלפים, ולכן נעשה נרמול בעזרת חלוקה ב- 1000.

בנוסף בעת יצירת הדוגמאות למרות שלקחתי בחשבון את השפעת תוצאות התיקו על הקבוצות (למשל לקחתי את זה בחשבון בעת יצירת ניקוד

ההיסטוריה של קבוצה), לא יצרתי אף דוגמא (לא לאימון ולא לחיזוי) שמכילה תוצאת תיקו, וזה כדי לצמצם את המרחב לבעייה בינארית (ניצחון\הפסד).

- **סיווג הדוגמאות :** לאחר הגדרת הבעית אין כל כך קושי בלסווג את הדוגמאות, מאחר ומידע זה כבר נמצא בתוך תוצאת המשחק, ולכן הוחלט לסמן :
  - 1 : אם קבוצה א" (כלומר הקבוצה המארחת תנצח)
  - 2 : אם קבוצה ב" (כלומר הקבוצה האורחת תנצח)
- **שלב הלמידה :** השתמשתי במספר אלגורתמים שונים, בכל אלגוריתם היתה המטרה לבנות מסווג, כך שבהינתן דוגמא (משחק) אפשר לסווג אותו(לחזות את התוצאה), בעזרת הלימדה שהאלגוריתם כבר עשה על דוגמאות שונות. בשלב זה המידע שמכיל את כל הדוגמאות המסווגות נמצא בקובץ processedGames.*csv* .

ברוב המקרים והנסויים שלב זה נעשה בשיטת *cross Validation* .

השימוש ב *crossValidation* היה זהה בכל האלגורתמים השונים כדי לנסות לנטרל שינויים בתוצאה הסופית (הדיוק בחיזוי) שיכולות לנבוע משינוי הפרמטרים ב- *crossValidation* , ולקבל הערכה יותר מדויקת של פעולת האלגוריתם עצמו.

בשיטה הנ"ל אוסף הדוגמאות היה מחולק ל- 10 קבוצות, עשיתי *shuffel* , והשתמשתי גם ב- $random\_state = 15161098$ (בכל מקום ובכל אלגוריתם) , שזה אומר שכל פעם שנעשה עירבוב ובכל מקומות השונים שעושים את זה יהיה אותו עירבוב ולכן אותה חלוקה.

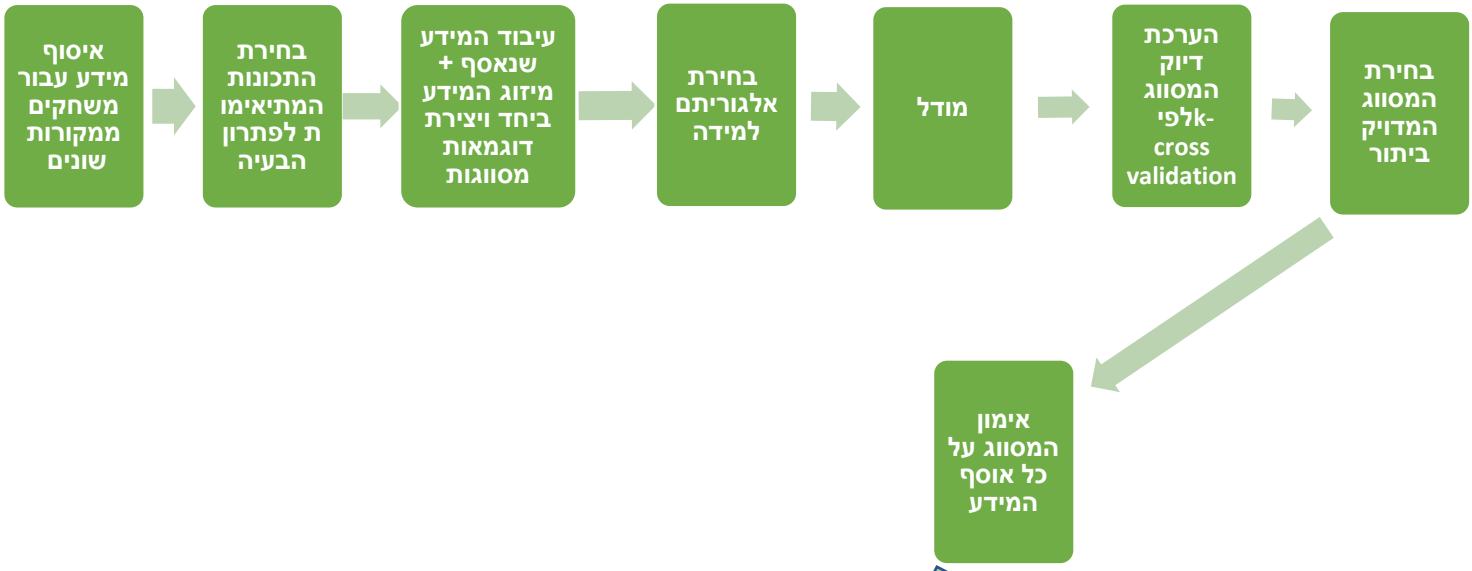
אחרי חלוקת ועירבוב ה- 10 קבוצות, יהיה 10 איטרציות שבכל פעם, תיבחר בדיוק קבוצה אחת שונה מבין ה- 10 להיות קבוצת הבדיקה (*test*), וכל תשעת הקבוצות האחרים יהיו דוגמאות הלימדה. אחרי הלימדה והחיזוי, בכל איטרציה יהיה הערכה (באחוזים) לכמה הצלחות היה בסיווג של קבוצת הטסט. אחוז ההצלחה הסופי יהיה הממוצע של אחוזי ההצלחה שהתקבלו ב- 10 ההרצות השונות. ואז המסווג הסופי יהיה המסווג שיש לו את ממוצע אחוז ההצלחה הכי גבוה מבין כולם.

המסווג הסופי שנבחר יאומן על כל אוסף המידע הקיים ויהיה מוכן לביצוע חיזוי.

בשלב זה השתמשתי ב 5 אלגורתמי *Machine Learning* שונים (פירוט בהמשך), עבור כל אלגוריתם עשיתי מספר ניסויים ביעקר על ערכי הפרמטרים שמוגדר בעזרתם המסווג.

- תיאור כללי של דרך הפתרון :

אימון :



חיזוי :

<u>תיאור המערכת :</u>

ביניתי מערכת שמורקבת ממספר סקרפטים ב Python 3 אשר מבצעים את כל מה שנאמר לעיל בחלק תיאור הפתרון המוצע.
המערכת מחולקת לתייקיות לפי המטרה של הסקרפטים, כמו למשל תייקית *classification* שמכילה את אלגורתמי הלמידה, ותייקית *classes* שמכילת את הסקרפטים ו- *classe*-ים שצריכים רעיבוד המידע.

- **עיבוד מקדים של הדוגמאות :** כפי שהסברתי קודם שלב זה הוא מערכת בפני עצמו, שמכיל בין היתר סימולציה של הרצת עונות הליגה הספרדית , כדי לקבל בכל נקודת זמן (כלומר לפני כל משחק) את המצב שהיה ברגע ההוא. כל עונה נעשה לה סימולציה ועיבוד בנפרד (כי התכונות של כל משחק מתקבלים במסגרת העונה שקורה בה המשחק), וגם בסוף כל המידע של כל העונות מתאחד לקבוץ דוגמאות מסווגות אחד, אין בעיה בלאחד דוגמאות של משחקים מעונות שונות, כי ברגע שיש לנו הדוגמאות המעובדות כל דוגמא תכיל את המידע הרלוונטי אליה ולא תלויה במידע שיש בדוגמאות אחרות.
- **שלב הלימדה והניסויים :**

שלב זה כולל את שלב הלמידה והרצת הניסויים. בשלב זה השתמשתי באלגורתמי הלמידה הבאים:

  - *Decision Tree*
  - *KNN*
  - *Extra Trees*
  - *Random Forest*
  - *SVM*

השימוש נעשה בעזרת חבילת $scikit-learn$.

את שלב הלמידה והנסויים, התבצע באופן הבא, עבור כל אחד מאלגוריתם הלמידה שהשתמשתי בהם יש *class* שכתבתי, בכל *class* עבור כל ניסוי יש פונקציה שמריצה אותו, באופן הבא :

  - טען את קובץ הדוגמאות שמכיל מידע מעובד
  - חלק את המאגר ל- 10 קבוצות שוות גודל, אחת לטסט, ו- 9 ללמידה ע"י $cross-validation$. (יש מספר קטן של ניסויים שנשעה בעזרת מספר הרצות של חלוקות רגילות של קבוצות למידה ומבחן, החלוקה נעשת בעזרת פונקציה מהספריה sklearn והתוצאה נלקחה כממוצע התוצאות בהרצות)

- צור את המסווג המתאים עם הפרמטרים המתאימים.
- בצע 10 איטרציות עם הפרמטרים שמתוארים בניסוי.

○ פלוט את התוצאה כגרף בתיקיית *results*.

בנוסף המערכת עצמה פולטת תמצית של הניסויים.

- **המסווג הסופי :**

המסווג הסופי הוא התוצאה של הניסויים. אחרי שהרצנו את כל הניסויים על כל האלגורתמים השונים שלנו, נבחר המסווג וגם הפרמטרים שלו שהניבו דיוק הכי גבוהה. ואז נעשה אימו של המסווג על כל המידע המעובד שיש לנו, כך שהמסווג מוכן לקבל וקטור שמתאר משחק ולסווג אותו, וזה נעשה בעזרת הקבצים בתוך תקיית *classification*, *input.csv* יכיל וקטור משחק (או יותר) לא מסווג שמתאר משחק, יסווג אותו ויפלוט את התוצאה בקובץ *output.csv*

.

- **אלרגורתמי הלמידה ופרמטריהם :**

בפרויקט שלי בחנתי מספר אלגורתמים שונים, כידוע אלגורתמי הלמידה יש להם חולשות וחזקות, אלגורתמים יכולים להיות טובים יותר מאלגורתמים שונים בפתירת אותה בעיה, בפרויקט הזה נבחן מספר אלגורתמים, ועבור כל אלגוריתם נעשה ניסויים בתקווה להשיג את ערכי הפרמטרים שמהווים לתוצאות הכי טובות (מבחינת דיוק המסווג).

הסבר קצר על כל אחד מאלגורתמי הלמידה שהשתמשתי בהם :

○ **KNN :**

אלגוריתם השכן הקרוב או k-Nearest Neighbors algorithm (או בקיצור k-NN), k-NN יכול לשמש לסיווג או לרגרסיה, בשני המקרים הקלט תלוי ב k-התצפיות הקרובות במרחב התכונות.
הקלט לשלב האימון של האלגוריתם הוא דוגמאות אימון, וקטורי תכונות במרחב רב ממדי כל אחד עם תווית סיווג (למשל עבור וקטור תכונות של d ממדים ושתי מחלקות סיווג). שלב האימון מתבסס רק על אחסון תכונת הווקטור ותווית הסיווג של דוגמאות האימון במבנה נתונים שיאפשר בהמשך חיפוש מהיר בהם, כדוגמת עץ kd.
הקלט לשלב הסיווג הוא וקטור ללא תווית. בשל הסיווג k מוגדר כקבוע, והמסווג קובע את תווית הסיווג על פי התווית השכיחה ביותר בקרב k דוגמאות האימון הקרובות לדוגמה הנבדקת.
אני בחנתי את הפרמטר :

n_neighbors: הפרמטר הזה מכריע איזה מספר של שכנים צריך לקחת בחשבון בזמן החלטה עבור דוגמא, כאשר מתרבלת דוגמא חדשה המודל יסתכל על n_neighbors השכנים הכי קרובים (לפי מטריקת מרחק כלשהי) כדי להכריע את הסיווג של הדוגמא.

- **SVM :**

מכונת תמך וקטורי (באנגלית Support Vector Machine) היא טכניקה של למידה מונחית המשמשת לניתוח נתונים לסיווג ולרגרסיה. דוגמאות האימון מיוצגות כווקטורים במרחב ליניארי. עבור בעיות סיווג, בשלב האימון מתאימים מסווג שמפריד נכון ככל האפשר בין דוגמאות אימון חיוביות ושליליות. כאשר מוצגת נקודה חדשה, האלגוריתם יזהה האם היא ממוקמת בתוך הקו המגדיר את הקבוצה, או מחוצה לו.
SVM אינו מוגבל רק לסיווג ליניארי, ויכול לבצע גם סיווג לא ליניארי.
אני בחנתי את הפרמטרים :

- פרמטר C : פרמטר רגולזציה. עוצמה הרגולזציה היא ביחס הפוך לערך של C.
- *Kernel*: סוג הקרנל שמשתמש בו המודל, יש כמה סוגים בינהם לינארי.

- **Extra Trees :**

Extra Trees (Extremely Randomized Trees) הוא אלגוריתם למידה שבונה מספר רק של עצי החלטה ומשלב את החיזוי שלהם כדי לבצע חיזוי. ההבדל העיקרי בין אלגוריתם זה לבין אלגרתמים אחרים מבוסיי עץ החלטה הוא האופן שבו העצים בנויים.
ב-Extra Trees, העצים נבנים באמצעות תת-קבוצה אקראית של התכונות בכל צומת וגם ספי הפיצול נבחרים באופן אקראי. האקראיות הזו בבחירת התכונה ובספי הפיצול הופכת את העצים לבלתי תלויים יותר זה בזה, מה שיכול להפחית את השונות ולהפוך את המודל לפחות נוטה להתאמת יתר.
אני בחנתי שני פרמטרים :

- max_depth: העומק המקסמאלי של העצים של המודל.
- n_estimators: מספר העצים שיבנה המודל.

- **Decision Tree :**

עץ החלטה (Decision Tree) הוא מודל חיזוי המספק מיפוי בין תצפיות לערכים המתאימים עבורן. עץ החלטה ממפה תצפיות על פריט ויוצר מסקנות על ערך היעד של הפריט. במבנה של עצים אלה, עלים

מייצגים סיווגים אפשריים וענפים מייצגים צירופים של תכונות אשר יובילו למחלקות הסיווג. בקבלת דוגמא חדשה לסיווג המודל ממפה את הדוגמא (לפי התכונות שלה) לעלה בעץ, ומחזיר את הסיווג שמוגדר בעלה כסיווג של הדוגמא שהתקבלה.
אני בחנתי שני פרמטרים :

- max_depth : העומק המקסמאלי של העץ של המודל.
- min_samples_leaf : המספר המינמאלי של דוגמאות שצריכות להיות בתוך עלה בעץ.

- **Random Forest :**

יער האקראי הוא אלגוריתם סיווג המורכב מעצי החלטה רבים, הוא משתמש ב- bagging וב- feature randomness בעת בנית כל עץ בודד בניסון ליצור יער של עצים לא מתואמים, טזה כדי ליצור "וועדה" שהחיזוי שלה ביחד יותר מדויק מזה של עץ בודד, בגלל שהיא מונעת התאמת יתר ומפחיתה רעשים.
אני בחנתי שני פרמטרים :

- max_depth : העומק המקסמאלי של העצים של המודל.
- n_estimators : מספר העצים שיבנה המודל.

ניסויים :

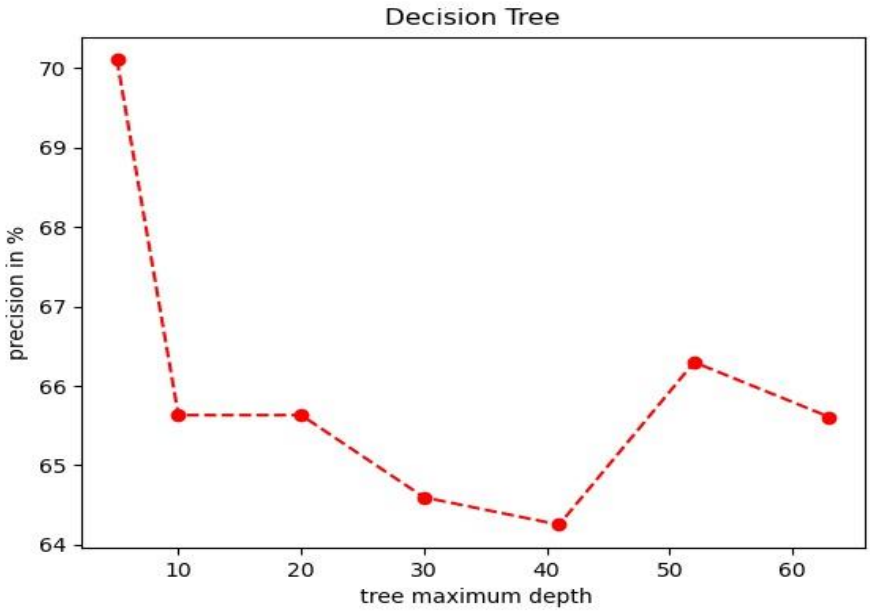
עשיתי מספר של ניסויים על מספר אלגורתמי למידה שונים, כדי לבחון איזה אלגוריתם למידה וגם איזה פרמטרים שלו הם הכי טובים (מבחינת דיוק) כדי ליצור מסווג שיפתור את הבעיה.

בכל אלגורתמי הלמידה עשיתי ניסוי נוסף שלא קשור בפרמטרים ליצירת המסווג, שזהו ניסוי שקשור לגודל קבוצת הטסט, ובדקתי את טיב המסווג שנוצר כקשר לאחוז קבוצה המבחן מאוסף הדוגמאות הקיים.

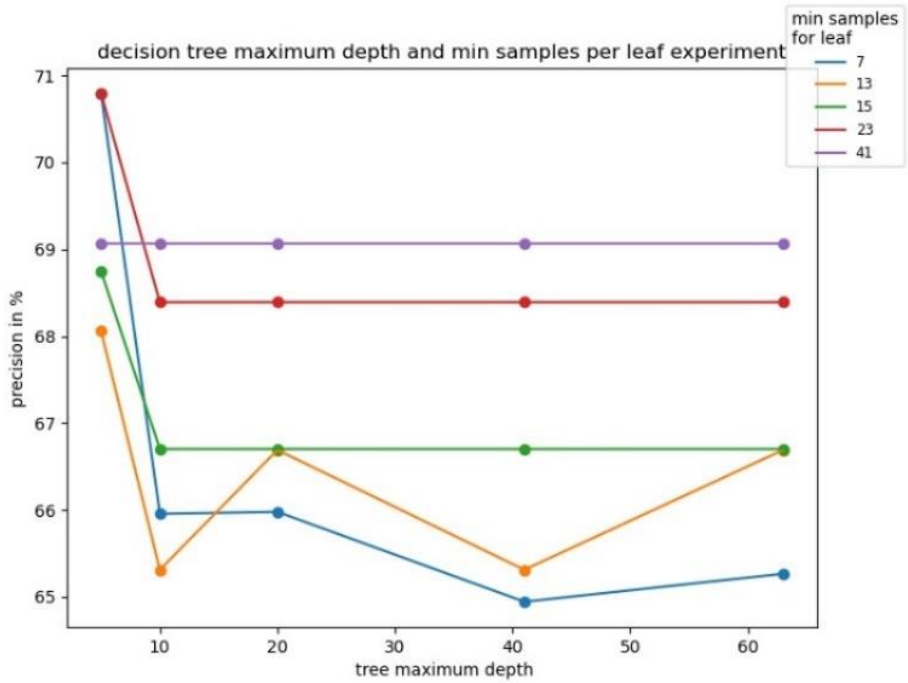
- **Decision Tree :**

בכל הניסויים הפרמטרים שלא נבדקים נקבעים לערך דיפולטיבי (שאני הגדרתי, רשימת הערכים הדיפולטיבים הם :
max_depth=None, splitter=best, test_size = 0.25
min_samples_leaf=7 ,criterion=entropy
בניסוי זה כמו שהזכרתי קודם בחנתי שני פרמטרים max_depth ו- min_samples_leaf . עשיתי כיוונון של הפרמטר max_depth לבד, ובניסוי נוסף עשיתי כוונון של שני הפרמטרים ביחד. והנה התוצאות :

- הגרף הבא מתאר את תוצאת הניסוי על פרמטר max_depth :

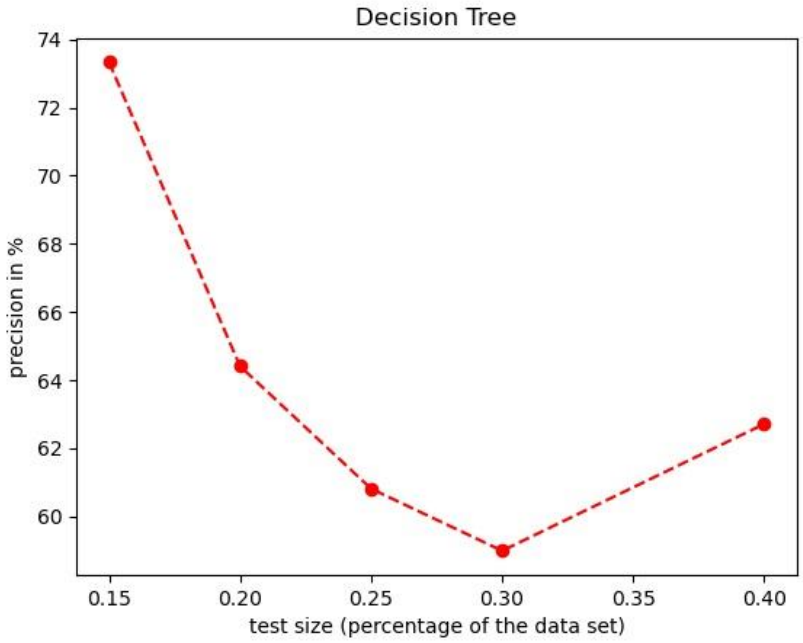


- תוצאת הניסוי של כוונון שני הפרמטרים max_depth ו-min_samples_leaf:



		tree maximum depth				
min samples for leaf		**5**	**10**	**20**	**41**	**63**
	7	70.79	65.95	65.98	64.94	65.26
	13	68.06	65.31	66.69	65.31	66.69
	15	68.75	66.7	66.7	66.7	66.7
	23	70.79	68.39	68.39	68.39	68.39
	41	69.07	69.07	69.07	69.07	69.07

- תוצאות הניסוי עבור גדול קבוצה הטסט :



אנו רואים כי בניסוי הראשון ערך ה- max_depth שהביא לתוצאה מקסמאלית הוא 7, וזה מעיד על כך שעמוקים גדולים יותר מ- 7 כנראה מהווה לתופעת overfitting , כי ככל שהעומק המקסמאלי הוא יותר גדול זאת אומרת שמסלולים בעץ יכולים ליות ארוכים יותר, כלומר יש חלוקה יותר מפורטת וקפדנית של קבוצה הדוגמאות כי כל צומת חדש במסלול מהווה איזושהי חלוקה של הדוגמאות.

עבור הניסוי השני יש כיוונון של שני פרמטרים ביחד, ניתן לראות שכמעט בכל הקווים יש אי-עליה בדיוק כאשר עולים בערך העומק המקסמאלי שזה גם מחזק מה שהסברנו עבור הניסוי הראשון.

בנוסף אנו מקבלים ערך דיוק מקסמאלי עבור

min_samples_leaf=7,max_depth=5 וגם עבור

min_samples_leaf=23,max_depth=5 , הפרמטר min_samples_leaf מהווה איזשהו מדד לטיפול ברעשים כאשר הוא יותר קטן זאת אומרת שמספר דוגמאות יותר קטן מכריע סיווג, וזה מעיד על כך שאין כל כך רעש בדוגמאות או שהרעש שקיים לא ממש משפיע על הסיווג, וזה המצב שיש לנו פה כי קיבלנו דיוק מקסמאלי בערכי min_samples_leaf יחסית קטנים.

בנוסף עבור הניסוי של גודל הטסט, קיבלנו את הדיוק הגדול ביותר עבור קבוצת הטסט הכי קטנה.

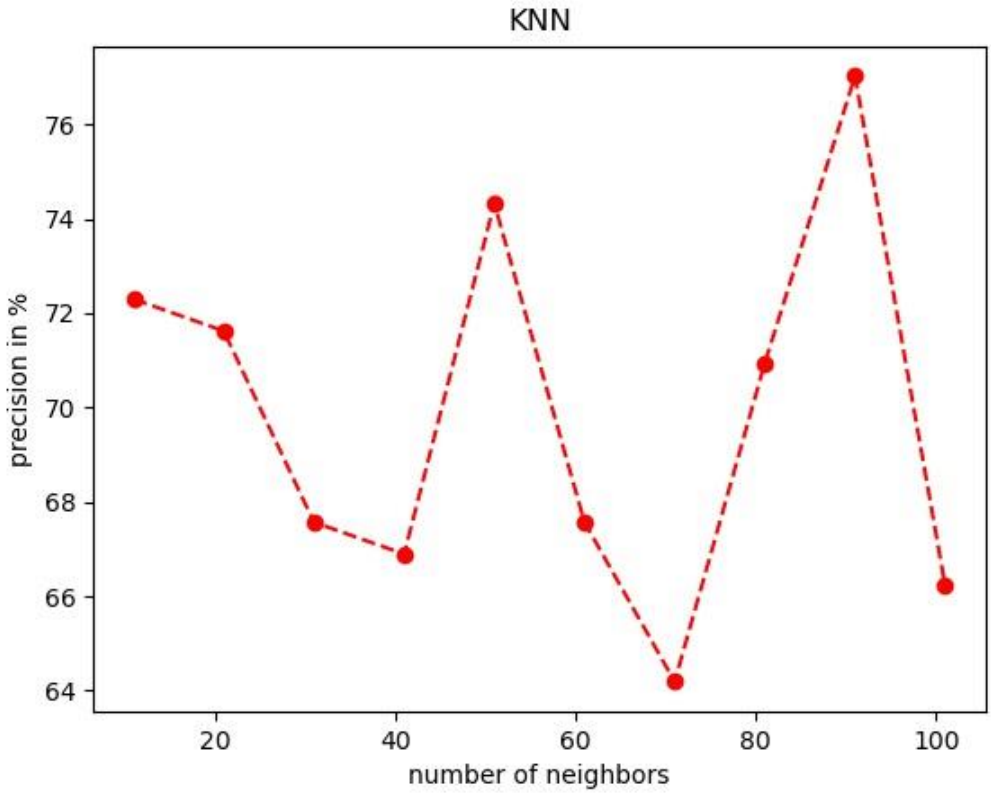
- **KNN :**

בכל הניסויים הפרמטרים שלא נבדקים נקבעים לערך דיפולטיבי (שאני הגדרתי), רשימת הערכים הדיפולטיבים הם :

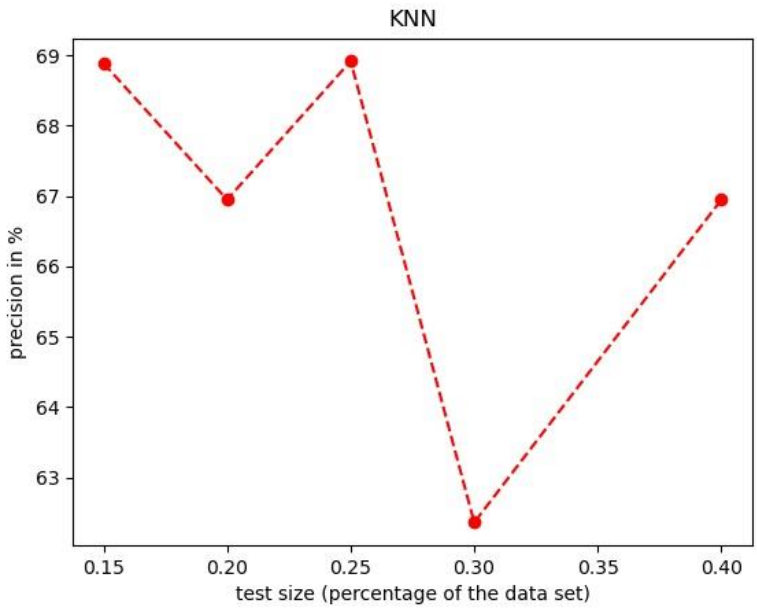
n_neighbors=5, weights="distance"

leaf_size=20

- בניסוי זה כמו שהזכרתי קודם בחנתי את השפעת ערך הפרמטר n_neighbors על הדיוק :



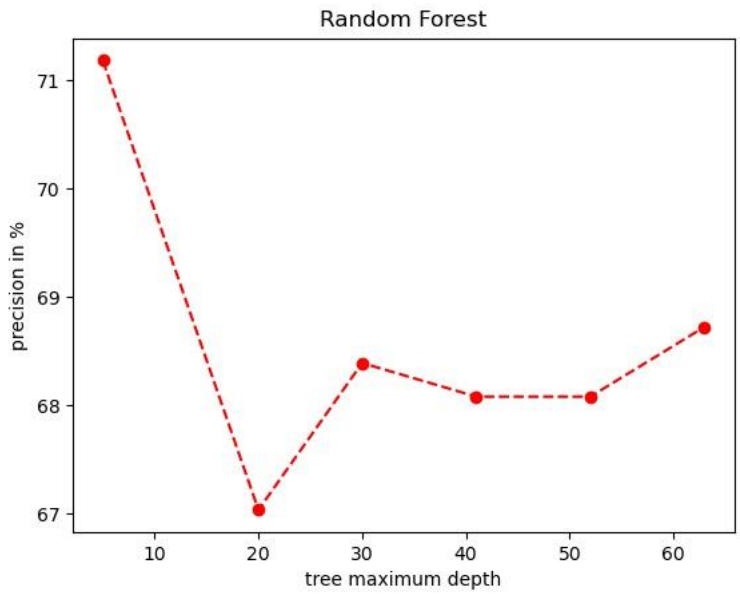
- תוצאות הניסוי עבור גדול קבוצה הטסט :



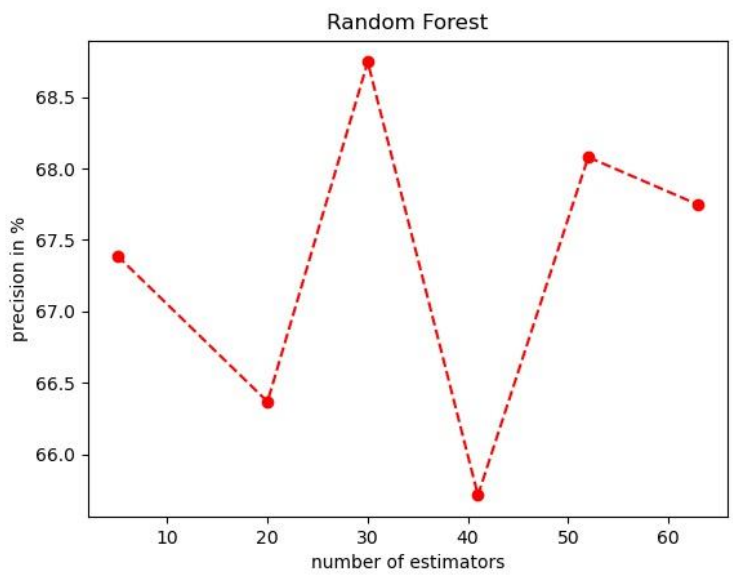
כפי שאפשר לראות הדיוק המקסמאלי מתקבל כאשר אנו משתמשים במספר יחסית גדול של שכנים, זה מעיד על כך שהדוגמאות לא נבדלות בקסלות, כלומר הבעיה לא קלה והסיווג לא קל, כי קבלת ההחלטה לא הייתה קלה והייתה מצטרכת מספר רב של שכנים כדי לקבל אותה.
עבור גודל קבוצת הטסט, באלגוריתם הזה הדיוקים הכי גבוהים היו עבור 15% וגם 25%.

- **Random Forest :**
  בכל הניסויים הפרמטרים שלא נבדקים נקבעים לערך דיפולטיבי לפי הרשימה הבאה :
  n_estimators=100, criterion='entropy',
  max_depth=None

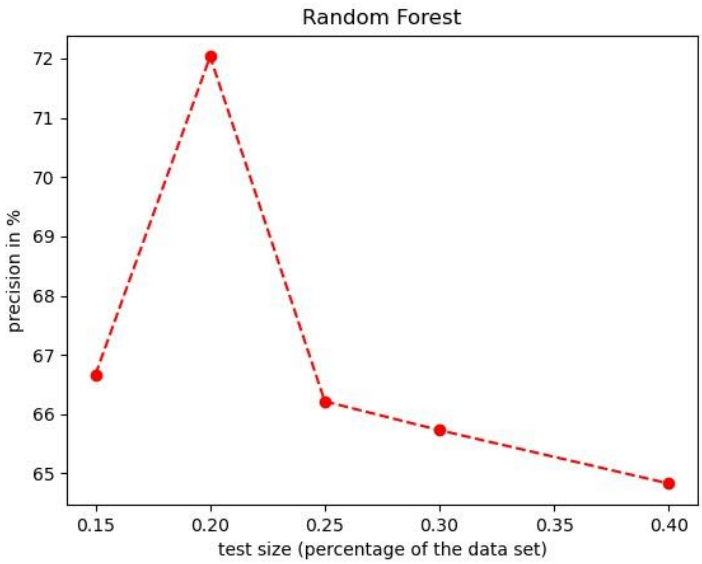
- תוצאות הניסוי עבור הפרמטר max_depth :



- תוצאות הניסוי עבור הפרמטר n_estimators:



- תוצאות הנהיסוי עבור גודל הטסט :



הדיוק כתלות בעומק המקסמאלי פה דומה לגרפים של אותו משתנה באלגורתמים אחרים, ואפשר לראות גם פה בציור כמו קודם שהעומק המקסמאלי הכי קטן מניב את הדיוק הכי גדול. כלומר אפשר להגיד שעומק גדול יותר מהווה לתופעת overfitting. בקשר למשתנה n_estimators, אנו יודעים שעם עלית מספר העצים המסווגים, הדיוק נוטה לעלות וההשפעה של רעשים נוטה לרדת, ובאותו זמן מספר גדול מדי של עצים מסווגים יכול להווה לתוספת overfitting על המידע של האימון וכתוצאה לפגוע בדיוק. לכן בגדול מצופה שערך זה יתחיל לעלות עד שיגיע למקסימום כלשהו, ואחר כך לרדת ולא להגיע למקסימוש זה שוב, זה מה שקיבלנו בניסוי שלנו, הערך שמהביא למקסימיום דיוק נמצא באמצע (לפי סדר) והוא 30, זה מעיד על כך שהאיזון ה- trade-off שהזכרנו לעיל מתקבל (אצלנו בפרוייקט) במספר 30.

בניסוי האחרון קיבלנו את הדיוק הטוב ביותר עבור קבוצת טסט בגדול 20%.

- **SVM :**

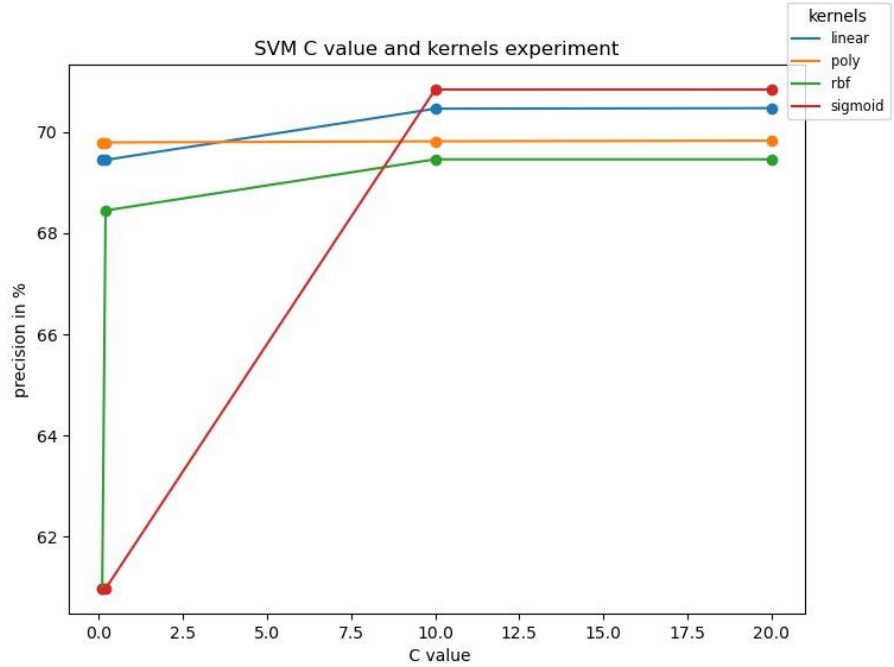
בכל הניסויים הפרמטרים שלא נבדקים נקבעים לערך דיפולטיבי לפי הרשימה הבאה :

kernel=rbf , C=1,

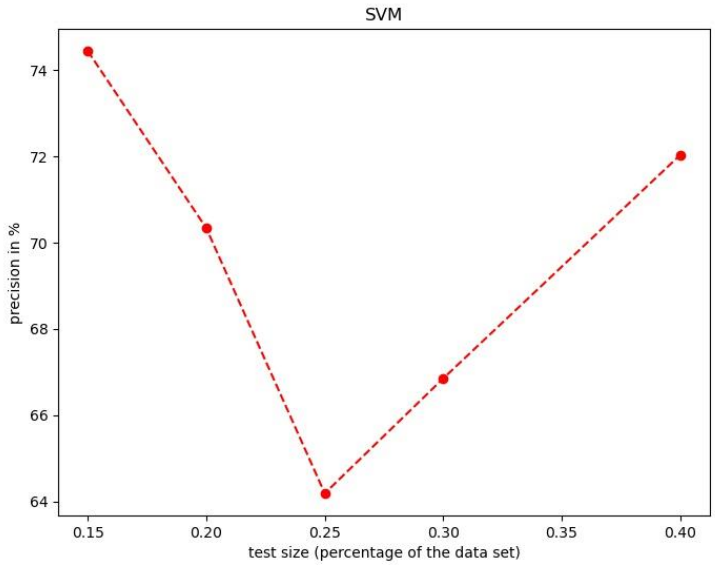
probability=True

  - תוצאת הניסוי של C ביחד עם kernel :

C value					
kernel		**0.1**	**0.2**	**10**	**20**
	linear	69.46	69.45	70.46	70.47
	poly	69.79	69.79	69.82	69.83
	Rbf	60.98	68.45	69.46	69.46
	sigmoid	60.98	60.98	70.84	70.84



- תוצאת הניסוי עבור ניסוי גודל הטסט :



ערך ה- C שולט בtrade-off בין מספר קטן של שגיאות אימון לבין מספר קטן של שגיאות מבחן, אפשר בבירור לראות שכמעט בכל הקרנילים השונים הדיוק עולה ככל שעולים בערך של C, כלומר כאשר מתענינים יותר בהפחתת שגיאות האימון. זה אומר שמסווג יותר כפדני פועל יותר טוב עבור הבעיה והנתונים האלו, אבל זה עד גבול מסויים מאחר שאחרי ערך של C ששווה ל-10 הדיוק ממשיך כמו שהוא ולא משתפר, וזה בגלל שבנסיון של "הקפדת" המסווג והקטנה שגיאת האימון יותר מדי אנו נכנסו למצב של overfitting ולכן לא רואים שיפור, כלומר ערך C האופטמאלי הוא 10.

אנו גם רואים שערך הדיוק המקסמאלי מתקבל אצל פונקצית קירנל *sigmond* , אנחנו מסיקים מכך (שעם גישה יותר קפדנית על שגיאות האימון) שמיפוי התכונות הכי טוב לבעיה הוא מיפוי לא לינארי , וזה מציע שמרחב התכונות הוא מרחב מסובך.

הדיוק הטוב ביותר פה בקשר לגודל קבוצה הטסט התקבל עבור 15%.

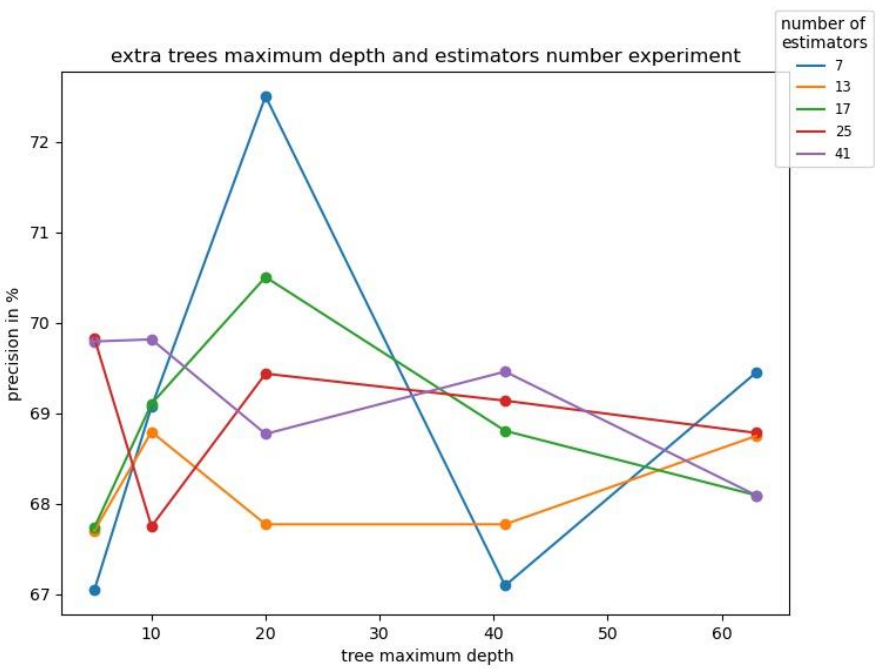
Extra Trees :

בכל הניסויים הפרמטרים שלא נבדקים נקבעים לערך דיפולטיבי לפי הרשימה הבאה :

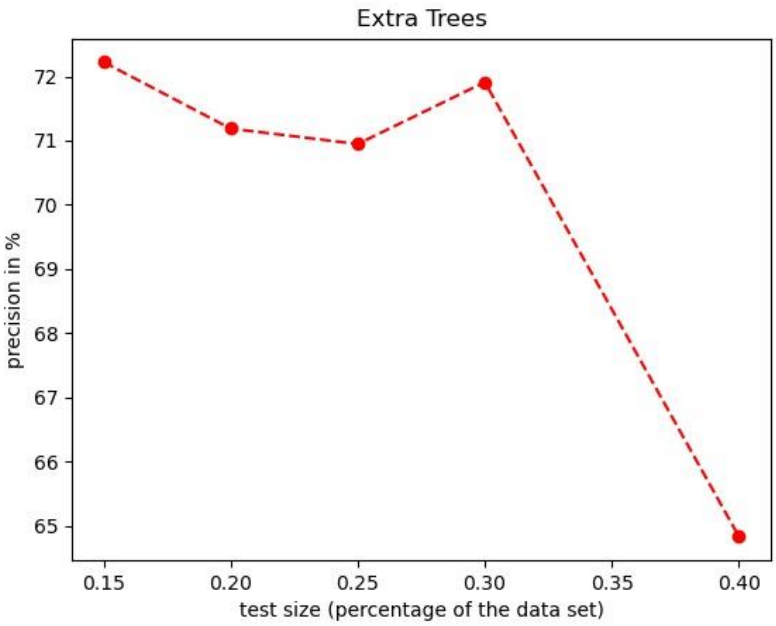
n_estimators=100 , max_depth=10,
min_samples_leaf=7, criterion="entropy", bootstrap=True

- תוצאת הניסוי של max_depth ו- n_estimators ביחד :

		tree maximum depth				
Number of estimators		**5**	**10**	**20**	**41**	**63**
	7	67.05	69.07	72.51	67.09	69.45
	13	67.7	68.79	67.77	67.77	68.75
	17	67.74	69.1	70.51	68.8	68.09
	25	69.83	67.75	69.44	69.14	68.78
	41	69.79	69.82	68.77	69.46	68.09



- תוצאת הניסוי על גודל הטסט :



מהניסוי הראשון אפשר לראות את החשיבות של כיוונון יותר מפרמטר ביחד, כי הפונקציה (הקו) של $n\_estimators = 7$ הוא זה שממקסם את הדיוק (במקרה הזה) למרות שעבור הרבה ערכי max_depth יש לו דיוק פחות משאר הקווים. וזה גם מעיד על כך שיש קשר לא מבוטל בין שני הפרמטרים.
הדיוק המקסמאלי שהתקבל בניסוי הוא 72.51% והוא התקסל בעומק 20 עם מספר מסווגים 7, אפשר לראות גם עבור ערכים קיצונים של הפרמטר max_depth
אין הרבה הבדלים בין הדיוק כתלות במספר המסווגים,

לדעתי זה נובע מכך שערכים מאוד גדולים/קטנים של max_depth מובילים לתופעת $underfitting/overfitting$ בהתאמה, עבור כל המסווגים, ולכן זה מוריד את השוני בין המסווגים השונים ומוריד את השפעת מספר המסווגים על הדיוק הסופי.
אפשר לראות גם עוד תופעות למשל לא משנה מהו העומק המקסמאלי מתקיים שהמסווג שמורכב מ- 13 מסווגים מניב את הדיוק הכי גרוע בניסוי, ומסיקים מזה שהערך 13 עבור n_estimators הוא ערך רע עבור מסווג ה- $Extra-Trees$ **ספציפית בבעיה הזו.**
כמו ברוב הניסויים האחרים על גודל קבוצת הטסט, הדיוק הטוב ביותר התקבל כאשר הגודל הוא 15%.

סיכום :

המטרה שלי הייתה ליצור פרויקט שיכול לתת חיזוי של תוצאות משחקי כדורגל, אשר ישמש למועדני כדורגל באנליזה ובהכנה לקראת המשחקים הבאים שלהם. בחרתי מספר אלגורתמי למידה ועשיתי הרבה ניסויים ברצון למצוא פתרון עם דיוק הכי גבוהה שאפשר. בסך הכל אלגוריתם KNN הביא לתוצאה הכי טובה, אפשר להסתכל גם על אלגוריתם $Extra-Trees$ שהביא לתוצאות לא רעות.

צריך לזכור שזו בעיה מאוד מורכבת ומסובכת, אנחנו יכולים לאוסף כמות עצומה של תכונות, ולבחור תכונות, אבל אף אחד לא מבטיח (גם אף אחד לא יודע) איזה פקטור ישפיע הכי הרבה על תוצאת המשחק הספיציפי הזה.וההוכחה של זה שאוהדי הכדורגל תמיד אומרים כי הכי טוב במשחקי כדורגל הוא שיש בו הרבה הפתעות. לכן אני יכול להגיד כי אחוזי הדיוק שמתוארים בפרוייקט הזה הם גבוהים בהינתן תיאור הבעיה.

במהלך העובדה נתקלתי במספר קשיים, למשל :

- בחירת התכונות היה דבר מאוד מאתגר, מבחינת והמדד של "מה משפיע תוצאת משחק כלשהוא ? " הוא לא מדד מוגדר היטב, והיה צורך בשימוש ביצרתיות וידע קודם בתחום הכדורגל כדי לפתור את הבעיה הזו.
- יש מידע חשוב שלא כתוב במפורש באיזשהוא מקור, אלא אפשר להסיק אותו מהמידע הקיים (למשל המומנטום והמצב הנפשי של השחקנים. שאפשר לקבל עליו מדד בעזרת התחשבות בתוצאות המשחקים האחרונים). דוגמא נוספת היא המיקום של כל קבוצה בטבלה בכל מחזור משחקים. בעיה זו נפתרה בעזרת מערכת שמבצעת סימלוציה של הוענות בליגה תוך עיבוד המידע.
- יש גם בתחום הזה ($machine-learning$) מספר לא קטן של אלגורתמי למידה, זה מהווה יתרון שהו באותו זמן גם חסרון, יש מבחר טוב ורחב של אלגורתמים, אבל זה גם מהווה אתגר לבחור איזה אלגוריתם לפתורון הבעיה. התגברתי על הקושי הזה בעזרת ניסויים רבים ששיקפו כמה כל אחד מהאלגוריתמים מתאים לפתרון הבעיה.

כיוונים עתידיים :

אפשר להרחיב או לשפר את הפקרוייקט הזה בהרבה צורות, חלק מהן :

- אפשר להסתכל על המצב הנפשי והאתלתי של כל שחקן בפני עצמו. קבוצה יכולה להיות במצב לא טוב ועם רצף רע של משחקים, אבל יכול להיות שיש לה שחקן שעובר בתקופה טובה ויכול "להציל" אותה ולשנות את התוצאות של מספר משחקים. מידע זה קשה מאוד לקבל, כי כדי להשיג אותו צריך לעשות אנליזה לרוב האירועים במשחקים האחרונים ולא רק לתוצאות שלהן, אבל לדעתי השיפור שיתווסף יהיה שווה את העבודה.

- אפשר לקחת בחשבון פקטורים נוספים שיכולים להשפיע על תוצאות המשחק, למשל אפשר להסתכל על זמן הנסיעה של הקבוצה האורחת, על מזג האוויר והשפעתו על כל אחת מהקבוצות, ועוד הברבה דברים...

- אפשר גם להרחיב את הפרוייקט ללמוד לחוזות תוצאות של תיקו, מעבר לזה אפשר לנסות לחזות תוצאה סופית שכוללת כמה גולים כל קבוצה תפקיע, אבל זו הרחבה מאוד גדולה ומסובכת שמצטרכת הרבה מחשבה וטיפול

ביבליוגרפיה :

[1] https://www.transfermarkt.com/laliga

[2] https://www.fifaindex.com

[3] https://www.flashscore.com

[4] https://www.wikipedia.org