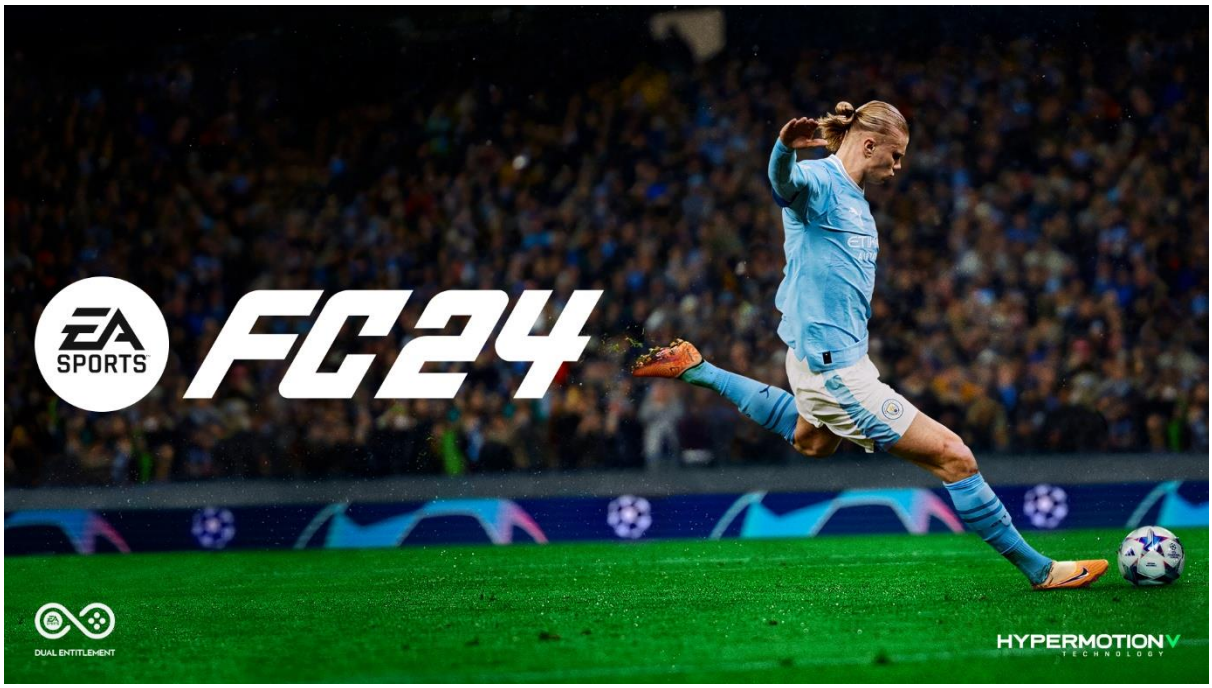
FIFA Player Stats Dataset

שם: יובל דהן

תעודת זהות: 208720383

קוד: https://github.com/Yuval10Dahan/Final-Project-ML



תיאור המאגר:

מאגר הנתונים מכילה סטטיסטיקות של שחקני FIFA (משחק הכדורגל למחשב ולקונסולות השונות) מעונות 2015 ועד 2024 כולל.

המאגר כולל נתונים כמו דירוג כולל, כישורים פיזיים וטכניים, שווי שוק ושכר, מידע על מועדונים ונבחרות שהשחקן שייך אליהם ועוד.

המאגר כולל תכונות עם מספרים ותכונות עם מחרוזות.

התכונות מתארות את ביצועי השחקנים, יכולותיהם והתקדמות הקריירה שלהם לאורך השנים.

מערכת נתונים זו מאפשרת ביצוע משימות שונות בלמידת מכונה, כגון חיזוי התפתחות שחקנים, סיווג עמדות וניתוח המאפיינים המרכזיים שמשפיעים על הביצועים.

כלים של למידת מכונה שהיה שימוש בהם במהלך הפרוייקט:

1) Linear Regression
2) Decision Tree
3) k-Nearest Neighbors (k-NN)
4) PCA
5) SVM
6) AdaBoost
7) Logistic Regression
8) Random Forest
9) Gradient Boosting

שאלה 1:

האם ניתן לחזות איך הדירוג הכולל (overall) של שחקן מסוים יתפתח במהלך 5 העונות הקרובות על סמך המאפיינים של השחקן בתחילת הקריירה שלו? (ב-3 הגרסאות הראשונות של משחק ה- FIFA שבהן הוא מופיע)

קישור לקוד המתאים: https://github.com/Yuval10Dahan/Final-Project-ML/tree/main/Q1

כלים שהרצתי:

1) **Linear Regression** – שימשה כמודל בסיסי לבדיקת קיום קשר ליניארי בין המאפיינים של השחקן בתחילת הקריירה שלו לבין הדירוג הכולל העתידי שלו ב-5 העונות הקרובות.
2) **Decision Tree** – נבדק עם עומקים שונים במטרה ללכוד דפוסים אפשריים שהם לא ליניאריים בנתונים עצמם.
3) **k-Nearest Neighbors (k-NN)** – נבדקו ערכים שונים של k כדי להשוות את ביצועי המודל מול מודלים אחרים.
4) **PCA** – מתוך 2 הרצות של הקוד, יישמתי פעם אחת Principal Component Analysis להפחתת המימדיות תוך כדי שמירה על 95% מהשונות (מידע) בנתונים.

תוצאות:

ביצעתי 2 הרצות של הקוד, ריצה אחת עם שימוש ב- PCA וריצה אחת ללא שימוש.

תוצאות ללא שימוש ב- PCA:

```
(.venv) PS C:\Users\yuval\Desktop\Code\PyCharm Code Projects\FinalProjectML> python predict_overall_rating.py
Dataset size: (46585, 17)


Linear Regression Results:
               Model  Mean Absolute Error  Root Mean Squared Error
0  Linear Regression             2.448463                  3.18617

Decision Tree Results:
                     Model  Mean Absolute Error  Root Mean Squared Error
0   Decision Tree (Depth=3)             2.645550                 3.418796
1   Decision Tree (Depth=5)             2.547863                 3.286060
2   Decision Tree (Depth=6)             2.614216                 3.386784
3   Decision Tree (Depth=7)             2.675273                 3.476380
4  Decision Tree (Depth=10)             2.991889                 3.996320

k-NN Results:
         Model  Mean Absolute Error  Root Mean Squared Error
0   k-NN (k=3)             3.227360                 4.129482
1   k-NN (k=5)             3.072293                 3.926904
2   k-NN (k=6)             3.040141                 3.886544
3   k-NN (k=7)             2.996422                 3.841249
4   k-NN (k=8)             2.986802                 3.825771
5  k-NN (k=10)             2.961079                 3.797866
Total runtime: 2.84 seconds
```

תוצאות עם שימוש ב- PCA:

```
(.venv) PS C:\Users\yuval\Desktop\Code\PyCharm Code Projects\FinalProjectML> python predict_overall_rating.py
Dataset size: (46585, 17)

Explained variance by PCA components: 0.95

Linear Regression Results:
                Model  Mean Absolute Error  Root Mean Squared Error
0  Linear Regression             2.849875                 3.649662

Decision Tree Results:
                      Model  Mean Absolute Error  Root Mean Squared Error
0   Decision Tree (Depth=3)             3.520681                 4.421400
1   Decision Tree (Depth=5)             3.366586                 4.256064
2   Decision Tree (Depth=6)             3.363853                 4.231703
3   Decision Tree (Depth=7)             3.322063                 4.198795
4  Decision Tree (Depth=10)             3.434094                 4.423045

k-NN Results:
         Model  Mean Absolute Error  Root Mean Squared Error
0   k-NN (k=3)             3.282338                 4.171618
1   k-NN (k=5)             3.121965                 3.990044
2   k-NN (k=6)             3.092357                 3.945684
3   k-NN (k=7)             3.068153                 3.918263
4   k-NN (k=8)             3.042702                 3.884815
5  k-NN (k=10)             3.037418                 3.872820
Total runtime: 2.36 seconds
```

המטרה של יישום PCA הייתה לבדוק האם הפחתה של מספר ה- features, תוך שמירה על 95% מהשונות, תשפר את הביצועים של המודל על ידי הסרה תכונות מיותרות או פחות שימושיות ("רעש").

Linear Regression:

Model	Without PCA / with PCA	Mean Absolute Error	Root Mean Squared Error
Linear Regression	Without PCA	2.448	3.186
	with PCA	2.850	3.649

רמת הביצועים ירדה כאשר הפעלתי PCA (עלייה בערכים של MAE ו- RMSE).

זה קורה בגלל ש- Linear Regression מסתמך על כך שכל התכונות שומרות על הקשרים שלהן במרחב המקורי. יישום ה- PCA יוצר "new transformed features", מה שעלול להסיר מידע חשוב שהיה שימושי במערך התכונות המקורי.

לכן ל- PCA הייתה השפעה שלילית על הרגרסיה הליניארית, כנראה בגלל שחלק מהתכונות היו מכריעות לתהליך החיזוי, אך שונו או אפילו הוסרו במהלך הפחתת המימדיות.

:Decision Tree

Model	Depth	Mean Absolute Error (Without PCA)	Root Mean Squared Error (Without PCA)	Mean Absolute Error (With PCA)	Root Mean Squared Error (With PCA)
Decision Tree	3	2.646	3.419	3.521	4.421
	5	**2.548**	**3.286**	3.366	4.256
	6	2.614	3.387	3.364	4.231
	7	2.675	3.476	**3.322**	**4.198**
	10	2.992	3.996	3.434	4.423

ללא שימוש ב-PCA המודל השיג ביצועים טובים יותר מאשר עם שימוש ב-PCA בכל עומק שנבדק.

עץ ההחלטה הטוב ביותר ללא שימוש ב-PCA הוא העץ בעל עומק ברמה 5 והוא השיג MAE של 2.548 בהשוואה לעץ ההחלטה הטוב ביותר עם שימוש ב-PCA, שהוא העץ בעל עומק ברמה 7, אשר השיג MAE של 3.322.

זה קורה בגלל שעצי החלטה, בדומה לרגרסיה ליניארית, מחלקים את מרחב התכונות על סמך הערכים המקוריים של התכונות. יישום ה- PCA יוצר "new transformed features", שאינן בהכרח מתיישרות עם האופן שבו עצים מחלקים נתונים, מה שמפחית את האפקטיביות שלהם.

לכן ל- PCA הייתה השפעה שלילית גם במקרה הזה על עץ ההחלטה.

:k-Nearest Neighbors (k-NN)

Model	k	Mean Absolute Error (Without PCA)	Root Mean Squared Error (Without PCA)	Mean Absolute Error (With PCA)	Root Mean Squared Error (With PCA)
k-Nearest Neighbors	3	3.227	4.129	3.282	4.171
	5	3.072	3.927	3.122	3.990
	6	3.040	3.887	3.093	3.946
	7	2.996	3.841	3.068	3.918
	8	2.986	3.826	3.042	3.885
	10	**2.961**	**3.798**	**3.037**	**3.873**

הביצועים של k-NN היו מעט גרועים יותר עם שימוש ב-PCA, אך ההבדל בין השימוש ב-PCA לחוסר השימוש ב-PCA לא חמור מדי כמו בעץ ההחלטה או ברגרסיה הליניארית.

המודל הטוב ביותר הוא המודל ללא שימוש ב-PCA בעל k = 10 אשר השיג MAE של 2.961, בעוד המודל עם השימוש ב-PCA הטוב ביותר בעל k = 10 גם כן והוא השיג MAE של 3.037.

זה קורה בגלל ש- k-NN מתבסס על מרחקים אוקלידיים, וכאשר PCA יוצר "new transformed features" , המרחקים בין הנקודות משתנים. למרות ש-PCA עשוי להסיר "רעש", הוא גם משנה מרחקים משמעותיים בין תכונות במימדים גבוהים, מה שמשפיע על ביצועי k-NN.

לכן ל-PCA הייתה השפעה שלילית קטנה אך מורגשת על k-NN, ככל הנראה מכיוון שהמרחב המקורי של התכונות סיפק מדדי מרחק משמעותיים יותר.

מסקנות:

1) PCA פגע באופן עקבי בביצועי המודלים השונים מה שמרמז על כך שכל התכונות המקוריות הכילו מידע שימושי.
2) Linear Regression היה המודל האופטימלי ביותר עם ביצועים טובים יותר ללא PCA.
3) Decision Tree נפגע הכי הרבה מהשימוש ב-PCA, בגלל ששיטות שמבוססות על עצים מסתמכות באופן רב על "structured feature splits".
4) K-NN הושפע פחות מהשימוש ב-PCA אך עדיין הציג ביצועים מעט נמוכים יותר עם שימוש ב-PCA.
5) ה-PCA הסיר מידע שהיה שימושי עבור המודלים.
במרחבים בעלי מימד גבוה, PCA יכול להסיר "רעש" אך במרחב הזה כל התכונות שנבחרו לפילטור ה-data היו חשובות.
6) PCA לא שיפר את התוצאות והגישה הטובה ביותר הייתה להשאיר את כל התכונות המקוריות ולהשתמש ברגרסיה ליניארית כמודל החיזוי היעיל ביותר.

PCA נדרש כאשר יש יותר מדי features (מעל 100), כאשר יש קורלציה גבוהה בין תכונות שמתקיימת על קבוצה גדולה של תכונות, וכאשר יש overfitting.

כיוון שהמאגר המסונן לא מכיל כמות גבוהה של features, PCA לא נדרש במקרה הזה.

מדוע החיזויים במודלים השונים כל כך קרובים?

1. רגרסיה ליניארית מנסה להתאים קו ישר בין ערכים מתחילת הקריירה (X) לערכים העתידיים (y).
במקרה הזה זה עובד היטב כי צמיחה של שחקן (דירוג כללי) היא לרוב ליניארית, במיוחד בשנים הראשונות של הקריירה עד לשיא שלו (בין 7 ל-10 שנים).
2. עצי החלטה מחלקים את ה-data לקבוצות דומות, ולכן כאשר יש יציבות במידע הקיים, מודלים בעומקים שונים עשויים לספק חיזויים דומים מאוד.
בנוסף, הדמיון בין החיזויים עשוי לנבוע מהעובדה שעומקי העצים שנבחרו אינם שונים באופן מהותי ב- decision structure שלהם. ייתכן שעץ בעומק 5 ועץ בעומק 7 מזהים את אותם דפוסים קריטיים, כך שהתוצאות הסופיות קרובות מאוד, עם הבדלים קטנים במדדי השגיאה.
3. K-NN מחפש את k השחקנים הדומים ביותר מגרסאות ה-FIFA הקודמות ולוקח את הממוצע של הדירוג הכללי העתידי שלהם כחיזוי.
הדמיון בחיזויים נובע מהעובדה שהדפוסים ההיסטוריים של השחקנים יציבים, כלומר מגמות העבר חוזרות על עצמן באופן עקבי בנתוני השחקנים החדשים.

מכיוון שמאגר הנתונים גדול ומגוון, מודלים שונים של למידת מכונה מגיעים לעתים קרובות למסקנות דומות לגבי התקדמות השחקנים, מה שמוביל ברוב המקרים להבדלים קטנים בחיזויים.

אתגרים:

1) **בחירת תכונות רלוונטיות:**

- תחילה השתמשתי בכל התכונות שבהן היו ערכים מספריים.
- לאחר שהתוצאות היו לא טובות, החלטתי לסנן באופן ידני ולהשאיר רק את התכונות הרלוונטיות לדעתי לשקלול דירוג כללי של שחקן (יש לי היכרות ברמה גבוהה עם כדורגל ואני חי את המקצוע הזה מגיל קטן).
- מדוע התכונות הללו עבדו טוב? תכונת ה- potential נותנת הערכה כמה גבוה שחקן יכול לצמוח בעתיד.
  תכונות פיזיות כמו גיל, גובה ומשקל משפיעות על התפקיד של השחקן במגרש ובהתאם לכך על הדירוג הכללי שלו (שחקני התקפה וקישור לרוב יהיו בעלי הדירוג הגבוה יותר).
  נתוני ביצועים כמו מהירות, דריבל, בעיטה וכו' עוקבים בצורה טובה אחר שיפור של שחקן, ולכן אם ישנו שחקן בעל צמיחה ביכולות השונות הללו לאורך השנים, ככה הסיכוי גבוה יותר לכך שהדירוג הכללי שלו יהיה גבוה יותר משמעותית.

2) **מציאת היפר-פרמטרים:**

- עצים עמוקים מדי ו-k נמוך מדי היו בעייתיים והיה צורך במציאת ערכים אופטימליים יותר.
- הרצה פשוטה על הערכים האפשריים שמוצגים בטבלאות ומציאה של הערכים האופטימליים מביניהם פתרה את הסוגייה הזאת.

3) **ערכים ריקים (Nans):**

- נתקלתי ב- 20,024 שורות של data שבהם היו ערכים ריקים, בגלל שלשוערים במשחק, בחלק מהתכונות הרלוונטיות יש ערכים ריקים כי אין רלוונטיות לתכונות האלה כשוער.
- אמנם זהו סינון של יחסית הרבה data אך זה לא פגע במודלים השונים ולכן מחקתי את שורות ה-data שמכילות שוערים.

טכניקות:

- עבדו טוב: Linear Regression

ה- Linear Regression עבד יותר טוב מבחינת Mean Absolute Error ו- Root Mean Squared Error.

מדוע? הקשר בין המאפיינים ההתחלתיים של השחקן לדירוג העתידי שלו הוא ברובו ליניארי, מה שהופך את הרגרסיה הליניארית להתאמה טובה.

כמו כן, קל להבין ולנתח את התרומה של כל תכונה לחיזוי.

- עבדו פחות טוב: Decision Trees, k-NN

ה- Decision Trees עלולים לסבול מ- overfitting במיוחד בעומקים גדולים.

הביצועים הטובים ביותר היו בעומק 5 אך הם לא עלו על ביצועי הרגרסיה הליניארית.

ה- k-NN השיג ביצועים טובים יותר מה- Decision Trees אבל עדיין לא עלה על הרגרסיה הליניארית.

שאלה 2:

האם ניתן לחזות את שווי השוק של שחקן על סמך הדירוג הכללי, הגיל והקבוצה אליה הוא שייך בלבד? האם יש להוסיף תכונות נוספות כדי למקסם את החיזוי? מחושב על כל גרסה של FIFA מ-2018 ועד 2024.

קישור לקוד המתאים: https://github.com/Yuval10Dahan/Final-Project-ML/tree/main/Q2

כלים שהרצתי:

5) **Linear Regression**
6) **Random Forest**
7) **SVM**
8) **AdaBoost**
9) **Decision Tree**
10) **k-Nearest Neighbors (k-NN)**

תוצאות:

תחילה התכונות שנבחרו היו רק הדירוג הכללי של אותו שחקן, הגיל והקבוצה אליה הוא שייך.

```
features = ['overall', 'age', 'club_team_id']
```

לפי הקוד אנחנו מבצעים חיזוי לכל גרסה של פיפא מ-2018 ועד 2024 בנפרד.

לאחר מכן מבצעים ממוצע על פני כל הגרסאות כדי לראות $r^2\ score$ ממוצע.

$r^2\ score$ מודד עד כמה מודל מסביר את השונות במשתנה המטרה שבמקרה שלנו הוא שווי השוק.

הנוסחה שלו היא:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

כאשר y_i הוא הערך האמיתי של שווי השוק עבור השחקן ה-i.

המשתנה $\hat{y}_i$ הוא הערך החזוי מהמודל עבור השחקן ה-i.

המשתנה $\bar{y}$ הוא הממוצע של כל הערכים האמיתיים y_i .

ציון קרוב ל-1 מעיד על חיזוי מעולה, ציון קרוב ל-0 או שלילי מעיד על חיזוי גרוע.

התוצאות שהתקבלו תחילה:

```
Average R² Scores Across All FIFA Versions:
Linear Regression: Average R² Score = 0.4251
Random Forest: Average R² Score = 0.9694
SVM: Average R² Score = -0.0945
AdaBoost: Average R² Score = 0.9093
Decision Tree: Average R² Score = 0.9421
KNN: Average R² Score = 0.9017
```

לאחר מכן הוספתי תכונות נוספות כדי לנסות לשפר את התוצאות של כל מודל.

התכונות לאחר ההוספה:

```
features = ['overall', 'potential', 'age', 'club_team_id', 'league_id',
            'club_contract_valid_until_year', 'nationality_id', 'release_clause_eur']
```

התוצאות שהתקבלו לאחר ההוספה:

```
Average R² Scores Across All FIFA Versions:
Linear Regression: Average R² Score = 0.9860
Random Forest: Average R² Score = 0.9896
SVM: Average R² Score = -0.0946
AdaBoost: Average R² Score = 0.9358
Decision Tree: Average R² Score = 0.9833
KNN: Average R² Score = 0.9799
```

ניתוח התוצאות:

1) Linear Regression בשלב הראשון נותן תוצאה גרועה.

סיבות לכך:

- שווי שוק של שחקן אינו עולה באופן ליניארי עם הדירוג הכולל או הגיל.
שחקן בן 20 עם דירוג 80 אינו שווה ערך כמו שחקן בן 28 עם אותו דירוג (השחקן בן ה-20 יהיה שווה יותר).
המודל הניח קשר ליניארי, אך קשר זה אינו מתקיים בעולם האמיתי של העברות שחקנים.
- שני שחקנים בעלי אותו דירוג כולל ואותו גיל יכולים להיות בעלי שווי שוק שונה לחלוטין.

הוספת תכונות כמו פוטנציאל, ליגה שבה השחקן משחק, אורך חוזה, לאום וסעיף שחרור ככל הנראה חשובים מאוד לחיזוי ערך שווי השוק של שחקן.

דוגמאות:

1. שחקן צעיר עם פוטנציאל גבוה וחוזה ארוך יהיה בעל שווי שוק גדול לעומת שחקן מבוגר שנשארה לו שנה אחת לסוף החוזה.
2. שחקנים ברזילאים או צרפתים לרוב בעלי שווי שוק גבוה יותר.
3. שחקן בעל סעיף שחרור של 100 מיליון יהיה שווה הרבה יותר משחקן עם סעיף שחרור של 10 מיליון. בלי התכונה הזאת המודל יתקשה ללמוד חוקים על מחירים של שחקנים.

השינוי המשמעותי (תוספת של 0.5609) ב- $r^2\ score$ במודל של Linear Regression מצביע על כך שרוב המידע שהיה חסר למודל נלכד על ידי הוספת התכונות החדשות האלו.

ממה נובע השינוי?

"פוטנציאל" מנבא את ההתפתחות העתידית של השחקן, מה שחשוב במיוחד עבור שחקנים צעירים.

"ליגה" ו- "לאום" מעידים על הביקוש בשוק ואת החשיפה של השחקן.

"סעיף שחרור" עוזר לקבוע את הערך המקסימלי של השחקן.

לכן הוספת תכונות אלו שיפרה את יכולת הרגרסיה הליניארית לחזות ערכים מציאותיים.

2) מודלים כמו Random Forest, AdaBoost, Decision Tree השתפרו לא באופן דרסטי.
סיבות לכך:
- מודלים מבוססי עצים תופסים באופן טבעי אי ליניאריות. אפילו רק עם 3 תכונות, המודלים הללו יכולים ללמוד חוקים מסובכים על שווי השוק. הוספת תכונות נוספות רק משנה מעט את החלוקות שלהם מאשר שהיא משפרת באופן דרסטי את הלמידה שלהם. דוגמה: עץ החלטה יכול ללמוד חוק בסגנון –
If overall > 85 and age < 23, then market value > 50M , הוספה של תכונה כמו פוטנציאל יכולה לעזור לשפר את החוק הזה אבל היא לא משנה באופן דרסטי את מבנה המודל.
- Random Forest ו- AdaBoost הם מודלים המשלבים מספר עצי החלטה. מודלים בסגנון הזה יכולים להפחית overfitting ולמצוא קשרים חזקים ב-data מוגבלת.

3) K-NN בשלב הראשון הניב ציון של 0.9017 בגלל שחישובי המרחק היו יותר פשוטים מאשר בשלב השני.
שחקנים עם גיל ודירוג כללי דומים אך עם שווי שוק שונה מאוד, נחשבו על ידי המודל ל"שחקנים קרובים", גם אם בפועל זה לא היה נכון.
בשלב השני k-NN הניב ציון של 0.9799 בגלל שהוספת התכונות נתנה סיפקה יותר דרכים להשוות שחקנים בדיוק רב יותר. זה עזר למודל לקבץ יחד שחקנים דומים יותר, מה שהוביל לחיזויים מדוייקים יותר.

4) אם $r^2\ score$ הוא שלילי, המשמעות היא שהמודל גרוע יותר מהבסיס והוא מספק חיזויים שמגבירים את השגיאה.
המשמעות של $r^2\ score$ שלילי עבור SVM היא שהמודל כנראה סובל מ- overfitting או underfitting.
מדוע SVM נותן ציון כל כך גרוע?
- חיזוי שווי שוק הוא בעיית רגרסיה ו-SVM אינו מתאים לכך.

- ערכי השוק בכדורגל אינם מתפלגים באופן אחיד. דוגמה: נניח ש-SVM לומד פונקציה שמתאימה לשחקנים בטווח מחירים בינוני (10-50 מיליון יורו). כאשר הוא נתקל בשחקן בשווי 150 מיליון יורו הוא יתקשה להכליל, מכיוון שהוא מנסה למצוא מישור הפרדה יחיד, מה שאינו מתאים לערכים קיצוניים.

מסקנות:

7) Random Forest השיג את הציון הגבוה ביותר מה שמעיד על כך שמודלים מבוססי עצים מתמודדים עם אי ליניאריות בצורה טובה יותר בהשוואה למודלים אחרים.
8) שווי שוק לא מקיים גבול החלטה פשוט ולכן SVM לא הצליח ללמוד שום דבר שימושי.
9) רגרסיה ליניארית צריכה את התכונות הנכונות כדי לעבוד כמו שצריך.
10) מודלים מבוססי מרחק צריכים מרחב תכונות עשיר כדי לבצע השוואות שימושיות.

אתגרים:

4) **הוספת תכונות** – תחילה המודלים עבדו על מרחב תכונות קטן יחסית והתוצאות היו פחות טובות ובחלק מהמודלים אפילו גרועות. לאחר הוספת תכונות חשובות ורלוונטיות המודלים התייצבו וסיפקו תוצאות מעולות ברובם.

טכניקות:

- עבדו טוב: Random Forest, Decision Tree, k-NN, AdaBoost, Linear Regression
מודל ה-Linear Regression עבד טוב בשלב השני לאחר הוספת התכונות החשובות.

- עבדו פחות טוב: SVM, Linear Regression
מודל ה-Linear Regression עבד לא טוב בכלל בשלב הראשון שכלל רק 3 תכונות.

שאלה 3:

האם ניתן לסווג שחקן למדינת המוצא שלו על סמך תכונותיו? אם לא, האם ניתן לסווג שחקנים מ-2 מדינות שונות במהותן בייצור שחקנים (כמו ברזיל לעומת גרמניה) למדינת המוצא שלהם על סמך תכונות היוצרות הבדל מהותי בין 2 הנבחרות?

קישור לקוד המתאים: https://github.com/Yuval10Dahan/Final-Project-ML/tree/main/Q3

כלים שהרצתי:

11) **Random Forest**
12) **Logistic Regression**
13) **SVM**
14) **Gradient Boosting** – מודל שבונה רצף של עצי החלטה, כאשר כל עץ חדש מתקן את השגיאות של העץ הקודם. לעתים מתעלה על Random Forest במאגרי נתונים מורכבים. עובד היטב עם נתונים לא מאוזנים על ידי התאמת משקלים באופן דינמי.

תוצאות:

תחילה חקרנו את השאלה על 10 הנבחרות הטובות בעולם לפי דירוג נוכחי של FIFA.

הנבחרות: ארגנטינה, צרפת, ספרד, אנגליה, ברזיל, פורטוגל, הולנד, בלגיה, איטליה, גרמניה.

התוצאות שהתקבלו:

```
Random Forest - Accuracy: 0.3219
```

```
Logistic Regression - Accuracy: 0.3157
```

```
Support Vector Machine - Accuracy: 0.3095
```

```
Gradient Boosting - Accuracy: 0.3315
```

התוצאות גרועות – אף מודל לא השיג יותר מ-33% דיוק.

סיבות לכך:

1) לאום של שחקן אינו יכול להיקבע אך ורק על פי מאפיינים של שחקני כדורגל.
2) שחקנים ממדינות שונות יכולים להיות בעלי מאפיינים דומים מאוד.
3) אין "דפוס כדורגל" ברור שמבדיל בין מדינות בדרך שקל ללמידת מכונה ללמוד, מצב כזה מקשה על כל מודל לזהות דפוסים משמעותיים.
4) לכמה מדינות יש הרבה יותר שחקנים מלשאר המדינות, לכן המודלים מוטים לכיוון המדינה הנפוצה ביותר ונוטים להיכשל בזיהוי המדינות הקטנות יותר. דוגמה מהנתונים:
נסתכל על המודל Gradient Boosting שלו יצא ה-accuracy הגבוה ביותר:

```
Gradient Boosting - Accuracy: 0.3315
Classification Report:
              precision    recall  f1-score   support

           7       0.18      0.07      0.10        42
          14       0.47      0.60      0.53       297
          18       0.24      0.16      0.19       179
          21       0.28      0.41      0.34       191
          27       0.15      0.07      0.10        97
          34       0.11      0.02      0.04        88
          38       0.11      0.01      0.02        84
          45       0.30      0.42      0.35       179
          52       0.33      0.49      0.39       159
          54       0.31      0.23      0.27       138

    accuracy                           0.33      1454
   macro avg       0.25      0.25      0.23      1454
weighted avg       0.29      0.33      0.30      1454
```

המודל נוטה לבצע overfitting למדינה הנפוצה ביותר (14) שלה יש 297 שחקנים.

המדינות הקטנות (7, 34, 38, 27) הן בעלות precision מאוד נמוך, משמע המודל מתקשה לזהות אותן נכון.

5) תכונות של שחקני FIFA מתמקדות בביצועים של שחקן, לא בלאום שלו. אלו לא התכונות הנכונות לסיווג של מדינה.
6) Multi-class classification קשה יותר מסיווג בינארי. מודלים כמו Logistic Regression ו-SVM מתקשים בסיווג כזה.
7) למדינות מסוימות יש סגנונות משחק חופפים. סיווג כל המדינות זאת משימה קשה, אך הגבלת הסיווג לסיווג בינארי בין 2 מדינות מנוגדות בסגנון המשחק עשויה לעבוד טוב יותר.

לכן שיניתי את דרך הפעולה והתמקדתי ב-2 מדינות בעלות סגנון משחק מנוגד – ברזיל לעומת גרמניה.

סגנון משחק ברזילאי מתאפיין לרוב בטכניות, יצירתיות, כדרור והתקפיות.

סגנון משחק גרמני מתאפיין לרוב בפיזיות, אינטילגנצייה טקטית, רב גוניות והתאמה למצבים שונים.

כמו כן שחקנים גרמניים נוטים להיות גבוהים יותר.

לפיכך נבחרו התכונות הללו:

```
features = ['height_cm', 'defending', 'attacking_finishing', 'skill_dribbling', 'movement_balance',
            "weak_foot", "passing", "dribbling", "physic", "attacking_short_passing", "skill_ball_control"
            "movement_sprint_speed", "movement_agility", "movement_reactions"]
```

התוצאות שהתקבלו לאחר בהשוואה בין גרמניה לברזיל:

```
Random Forest - Accuracy: 0.8464
```

```
Logistic Regression - Accuracy: 0.7831
```

```
Support Vector Machine - Accuracy: 0.7831
```

```
Gradient Boosting - Accuracy: 0.8434
```

הביצועים של המודלים השתפרו באופן משמעותי.

סיבות לכך:

1) שחקנים מ-2 המדינות הללו מספקים הפרדה מספיק טובה בתכונות של FIFA וזה מאפשר למודלים לסווג אותם בצורה טובה יותר.
2) סיווג בינארי הוא משמעותית פשוט יותר מסיווג multi-class. המודלים צריכים למצוא רק גבול החלטה אחד במקום מספר גבולות החלטה.
3) לגרמניה וברזיל יש מאגר נתונים מאוזן יחסית (205 שחקנים גרמניים, 127 שחקנים ברזילאיים), זה מונע הטייה כלפי קבוצה אחת ומאפשר סיווג הוגן יותר.

הביצועים הכי טובים היו של Random Forest בגלל שהוא מצליח ללכוד היטב אי ליניאריות.

Gradient Boosting עם ביצועים דומים מאוד בגלל שהוא משפר את ביצועי עצי ההחלטה צעד אחר צעד.

LR ו-SVM השיגו ביצועים נמוכים יותר בגלל שהם מניחים הפרדה ליניארית, שאינה אידיאלית עבור מאגר הנתונים הנ"ל.

```
Random Forest - Accuracy: 0.8464
Classification Report:
              precision    recall  f1-score   support

          21       0.85      0.91      0.88       205
          54       0.84      0.74      0.79       127

    accuracy                           0.85       332
   macro avg       0.84      0.83      0.83       332
weighted avg       0.85      0.85      0.84       332
```

נסתכל על Random Forest לעומק:

לברזיל יש recall גבוה יותר – המודל הצליח לזהות את רוב השחקנים הברזילאים בצורה נכונה.

גרמניה עם recall נמוך יותר – יותר שחקנים גרמנים סווגו בטעות כברזילאים.

ה-precision היה דומה ל-2 המדינות, מה שאומר שהמודל לא היה מוטה כלפי מדינה מסוימת.

הנתונים הללו מרמזים שלשחקנים ברזילאים יש הבדלים סטטיסטיים ברורים יותר, בעוד שחלק מהשחקנים הגרמנים חולקים מאפיינים עם סגנון ברזילאי.

מסקנות:

11) מודלים מבוססי עצים הם הבחירה הטובה יותר עבור משימת הסיווג הזו.
12) LR ו-SVM אינם אידיאליים עבור סיווג אי ליניארי.

אתגרים:

5) **סיווג multi-class** – סיווג שנעשה על 10 מדינות וגרם לתוצאות גרועות. לאחר מכן נעשתה התמקדות ב-2 נבחרות עם סגנון משחק שונה והתוצאות השתפרו בצורה משמעותית.
6) **מאגר לא מאוזן** – למדינות מסוימות היו הרבה יותר שחקנים מלשאר המדינות. התמקדות ב-2 נבחרות בעלות מספר דומה של שחקנים איזנה את הנתונים.
7) **חפיפה בין תכונות** – תכונות שחקנים כגון מסירה, כדרור ועוד משותפות בין מדינות שונות. המודלים לא הצליחו להבדיל בין מדינות מכיוון שסגנונות המשחק שלהם חפפו.
8) **סיווג בין מדינות הוא לא ליניארי** – מודלים מסויימים כמו LR ו-SVM התקשו מפני שהם מניחים שקיימת הפרדה ליניארית.

טכניקות:

- עבדו טוב: Random Forest, Gradient Boosting
- עבדו פחות טוב: Logistic Regression, SVM

המודלים הללו פשוטים מדי לסיווג לפי מדינה, התקשו להתמודד עם אי ליניאריות.

שאלה 4:

האם למידת מכונה יכולה למיין שחקנים לתפקידים לפי התכונות שלהם?

קישור לקוד המתאים: https://github.com/Yuval10Dahan/Final-Project-ML/tree/main/Q4

כלים שהרצתי:

15) **SVM**
16) **Decision Tree**
17) **Random Forest**
18) **Logistic Regression**

תוצאות:

מאגר הנתונים בשאלה הזאת הוא multi-label, כלומר שחקן יכול להיות עם כמה תפקידים.

Precision – מודד כמה מתוך השחקנים שהמודל סיווג לעמדה מסוימת אכן שייכים לה.

Recall – מודד כמה מתוך השחקנים ששייכים בפועל לעמדה מסוימת זוהו נכון על ידי המודל.

F1-score – ממוצע הרמוני של Precision ו- Recall, מעין מדד מאוזן לביצועי המודל.

תחילה המודלים אומנו על כל התפקידים במשחק. (15 תפקידים בסה"כ)

חלק מהתפקידים הכילו נתונים לא מאוזנים ולא מספיקים.

```
SVM Accuracy: 0.8168
              precision    recall  f1-score   support

         CAM       0.33      0.89      0.48       428
          CB       0.78      0.93      0.85       779
         CDM       0.49      0.91      0.64       554
          CF       0.07      0.89      0.13        90
          CM       0.67      0.92      0.78       827
          GK       1.00      1.00      1.00       402
          LB       0.38      0.91      0.53       404
          LM       0.32      0.86      0.47       480
          LW       0.21      0.91      0.34       228
         LWB       0.15      0.88      0.25       155
          RB       0.32      0.91      0.48       378
          RM       0.29      0.83      0.43       468
          RW       0.20      0.91      0.33       228
         RWB       0.15      0.91      0.26       160
          ST       0.74      0.91      0.82       643
```

תפקידים כמו CF, LWB וכו הכילו ממש מעט data.

כמו כן, ניתן לראות שהתפקיד של השוער (GK) סיפק חיזוי מושלם, ככל הנראה בגלל שחלק מהתכונות שנבדקו מכילות נתונים נמוכים מאוד ביחס לשוערים.

סיננתי את התפקידים ככה שיהיו רק תפקידים שמכילים data הגבוהה מ-400.

קיבלתי את התוצאות הבאות:

Decision Tree Accuracy: 0.8906

	precision	recall	f1-score	support
CB	0.90	0.81	0.85	811
CM	0.70	0.69	0.70	852
ST	0.87	0.79	0.83	629
LB	0.42	0.25	0.31	361
CDM	0.62	0.56	0.59	597
RM	0.40	0.19	0.26	444
LM	0.40	0.24	0.30	455
CAM	0.54	0.46	0.50	436
micro avg	0.68	0.56	0.62	4585
macro avg	0.61	0.50	0.54	4585
weighted avg	0.65	0.56	0.60	4585
samples avg	0.53	0.50	0.50	4585

SVM Accuracy: 0.8689

	precision	recall	f1-score	support
CB	0.81	0.91	0.86	811
CM	0.74	0.88	0.81	852
ST	0.77	0.90	0.83	629
LB	0.39	0.81	0.53	361
CDM	0.61	0.89	0.73	597
RM	0.33	0.75	0.46	444
LM	0.35	0.75	0.48	455
CAM	0.42	0.76	0.54	436
micro avg	0.55	0.85	0.67	4585
macro avg	0.55	0.83	0.65	4585
weighted avg	0.60	0.85	0.70	4585
samples avg	0.55	0.72	0.60	4585

Logistic Regression Accuracy: 0.8401

	precision	recall	f1-score	support
CB	0.79	0.93	0.85	811
CM	0.67	0.91	0.77	852
ST	0.74	0.93	0.82	629
LB	0.36	0.90	0.52	361
CDM	0.52	0.91	0.66	597
RM	0.30	0.83	0.44	444
LM	0.33	0.85	0.47	455
CAM	0.35	0.87	0.50	436
micro avg	0.49	0.90	0.64	4585
macro avg	0.51	0.89	0.63	4585
weighted avg	0.56	0.90	0.67	4585
samples avg	0.51	0.75	0.58	4585

Random Forest Accuracy: 0.9076

	precision	recall	f1-score	support
CB	0.93	0.82	0.87	811
CM	0.84	0.67	0.75	852
ST	0.92	0.79	0.85	629
LB	0.60	0.14	0.23	361
CDM	0.80	0.55	0.66	597
RM	0.47	0.04	0.07	444
LM	0.56	0.09	0.16	455
CAM	0.72	0.25	0.37	436
micro avg	0.85	0.50	0.63	4585
macro avg	0.73	0.42	0.49	4585
weighted avg	0.77	0.50	0.57	4585
samples avg	0.52	0.46	0.47	4585

	SVM	Decision Tree	Random Forest	Logistic Regression
Accuracy	0.868869	0.890565	0.907561	0.840123

ניתוח נתונים:

1) בחלק מהתפקידים ה- precision, recall נמוכים מאוד.
לדוגמה: ב- Decision Tree הציונים של התפקיד LB (מגן שמאלי) היו 0.42 ו-0.25 בהתאמה.
זה בגלל שהמודל יכול להתבלבל בין מגנים לבלמים (CB) או קשרים אחוריים (CDM), מאחר והם חופפים בתכונות שלהם כמו תיקולים וסיבולת.
2) לתפקידים כמו RM, LM, CAM יש f1-score נמוך מאוד בכל המודלים.
זה בגלל שהתפקידים הללו חופפים בתכונות שלהם כמו דריבל ומסירה.
זה מקשה על המודלים להפריד ביניהם.
3) לתפקיד ה-CDM (קשר אחורי) יש Precision נמוך ברוב המודלים.
ככל הנראה כי המודל מתבלבל בינו לבין בלם (CB), מגן (LB) או קשר אמצע (CM).

לכן כדי לשפר את הנתונים החלטתי לחלק את התפקידים במשחק ל-3 קטגוריות כלליות:

הגנה – לכן בחרתי בתפקיד הבלם (CB), לו יש data הכי גבוהה משחקני ההגנה.

קישור – לכן בחרתי בתפקיד הקשר אמצע (CM), לו יש data הכי גבוהה משחקני הקישור.

התקפה – לכן בחרתי בתפקיד החלוץ (ST), לו יש data הכי גבוהה משחקני ההתקפה.

אלו התוצאות שקיבלתי:

Decision Tree Accuracy: 0.9144

	precision	recall	f1-score	support
CB	0.88	0.82	0.85	776
CM	0.70	0.70	0.70	813
ST	0.87	0.79	0.83	675
micro avg	0.81	0.77	0.79	2264
macro avg	0.82	0.77	0.79	2264
weighted avg	0.81	0.77	0.79	2264
samples avg	0.47	0.47	0.46	2264

SVM Accuracy: 0.9256

	precision	recall	f1-score	support
CB	0.82	0.91	0.86	776
CM	0.75	0.89	0.81	813
ST	0.75	0.91	0.82	675
micro avg	0.77	0.90	0.83	2264
macro avg	0.77	0.90	0.83	2264
weighted avg	0.78	0.90	0.83	2264
samples avg	0.54	0.54	0.54	2264

Logistic Regression Accuracy: 0.9116

	precision	recall	f1-score	support
CB	0.79	0.93	0.85	776
CM	0.67	0.91	0.77	813
ST	0.72	0.92	0.81	675
micro avg	0.72	0.92	0.81	2264
macro avg	0.73	0.92	0.81	2264
weighted avg	0.73	0.92	0.81	2264
samples avg	0.54	0.56	0.54	2264

Random Forest Accuracy: 0.9313

	precision	recall	f1-score	support
CB	0.92	0.83	0.87	776
CM	0.83	0.69	0.75	813
ST	0.92	0.77	0.84	675
micro avg	0.89	0.76	0.82	2264
macro avg	0.89	0.76	0.82	2264
weighted avg	0.89	0.76	0.82	2264
samples avg	0.47	0.46	0.46	2264

	SVM	Decision Tree	Random Forest	Logistic Regression
Accuracy	0.925613	0.914351	0.931335	0.911626

ניתוח נתונים:

1) תפקיד הקשר אמצע (CM) עם Recall סביר במודלים כמו Decision Tree ו-Random Forest.
כמו כן, הוא עם Recall גבוה מאוד במודלים כמו SVM ו-Logistic Regression.
זה ככל הנראה בגלל שהוא לא מושווה לתפקידים דומים כמו CDM, LM וכו'.
2) תפקיד הבלם (CB) עם נתונים גבוהים (מעל 0.79) בכל המודלים.
3) תפקיד החלוץ (ST) עדיין עם נתונים טובים גם לאחר ההורדה של תפקידי התקפה שיכולים לחפוף בנתונים כמו RM,LM .

מסקנות:

13) **SVM** –

- מתאים להפרדה ברורה של תפקידים כמו בלם, קשר אמצע וחלוץ.
- פחות אידיאלי להפרדת תפקידים שבהם יש חפיפה בתכונות כמו קשר אמצע (CM) מול קשר התקפי (CAM).

14) **Decision Tree** –

- עובד בצורה טובה יותר כאשר יש חוקים ברורים להפרדה בין תפקידים כמו בלם מול חלוץ.
- למודל היה קושי בזיהוי תפקידים מורכבים כמו CAM, RM, LM בניסיון הראשון.
- המודל השתפר בניסיון השני אך הושגו ביצועים טובים יותר עם Random Forest.

15) **Random Forest** –

- המודל עם ה-accuracy הכי טוב מכל המודלים.
- גם כאשר היו הרבה תפקידים, עדיין הציג ביצועים טובים יותר לעומת מודלים אחרים.

16) **Logistic Regression** –

- אינו מתאים כאשר יש הרבה תפקידים שחופפים בתכונות.
- הציג ביצועים הרבה יותר טובים בניסיון השני, מה שמראה שהוא מתפקד היטב כאשר יש קטגוריות ברורות להפרדה.

אתגרים:

9) **נתונים לא מאוזנים** – כאשר כל התפקידים במשחק נכללו היו לנו נתונים לא מאוזנים, חלק מהתפקידים היו עם נתונים גבוהים וחלק היו עם נתונים כמעט אפסיים.
הפתרון היה לבחור רק את התפקידים שמכילים data גבוהה מ-400.

10) **ערכים נומריים** – תחילה כללתי את כל התכונות שיש להן ערכים מספריים. התוצאות היו פחות טובות כי היו תכונות שכללו מידע מספרי שאינו רלוונטי לחיזוי.
לדוגמה: תכונות כמו fifa_version או fifa_update.
הפתרון היה לסנן באופן ידני את התכונות הלא רלוונטיות.

11) **תפקיד השוער** – שוערים סיפקו חיזוי מושלם.
הפתרון היה לסנן שוערים לגמרי.

12) **שחקן עם יותר מתפקיד אחד** – ישנם שחקנים עם כמה תפקידים.
הפתרון היה לחלץ את כל התפקידים במשחק ולהמיר את הרשימה הזאת למטריצה בינארית אשר מקיימת:

- שורות מייצגות שחקנים.
- עמודות מייצגות תפקידים.
- לכל שחקן יש 1 בעמודה שבה ישנו תפקיד שהוא ממלא ו-0 בעמודה שבה יש תפקיד שלא קשור אליו.

טכניקות:

- עבדו טוב: Random Forest, SVM

Random Forest –
המודל הטוב ביותר.
משתמש בכמה עצי החלטה, מה שמפחית overfitting.
גם בניסיון הראשון כשהמודל קצת התקשה עם תפקידים מסוימים, הוא עדיין סיפק ביצועים טובים יותר משאר המודלים.
SVM –
המודל טוב בלהפריד תפקידים שונים במהותם.
המודל הצליח לזהות נכון את רוב השחקנים בכל תפקיד (Recall גבוה).

- עבדו פחות טוב: Decision Tree, Logistic Regression

Decision Tree –
המודל גרם ל-overfitting כאשר היו מספר רב של תפקידים (ציון Recall גבוה ל-CB ו-ST לעומת ציון Recall נמוך ל- RM, LM – משמע המודל "שינן" תפקידים ברורים במהותם אבל התקשה להכליל קשרים כמו שצריך).
Recall נמוך יותר בהשוואה ל-SVM ול- Random Forest, משמע המודל פספס סיווגים נכונים רבים.
התקשה עם תפקידים חופפים כמו CM, CDM, CAM.
Logistic Regression –
המודל ליניארי והוא התמודד עם משימה מורכבת ולא ליניארית.
המודל התקשה עם תפקידים חופפים ולעתים קרובות סיווג אותם בצורה שגויה.
המודל הכליל בצורה גרועה עם מספר רב של תפקידים.

שאלה 5:

האם ניתן לחזות אילו שחקנים צעירים (מתחת לגיל 20) החל מ- FIFA 2021 ומטה יגיעו לדירוג כולל של מעל 85 ב- FIFA 2024, בהתבסס על המאפיינים שלהם מתחילת הקריירה?

קישור לקוד המתאים: https://github.com/Yuval10Dahan/Final-Project-ML/tree/main/Q5

כלים שהרצתי:

19) **Decision Tree** – אחד מהמודלים בעלי הביצועים הגרועים ביותר, ככל הנראה בשל overfitting לנתוני האימון. כמו כן, העץ התקשה להכליל היטב (generalize) על שחקנים חדשים.

20) **SVM** – המודל עם הביצועים השניים הטובים ביותר מבחינת שגיאה (MAE = 4.436), מעט מאחורי AdaBoost, ככל הנראה בגלל שהמפריד הליניארי היה סביר והמודל הצליח להימנע מ- overfitting בצורה טובה יותר מ- Decision Tree.

21) **AdaBoost** – הביצועים הטובים ביותר מבחינת שגיאה (MAE = 4.314). ככל הנראה בגלל שהוא מתקן את בעיית ה-overfitting של ה-Decision Tree באמצעות חיזוק "weak learners".

22) **Logistic Regression** – ביצועים טובים, יותר מ- Decision Tree ומ- k-NN, ככל הנראה משום שהנתונים הראו מגמות ליניאריות מסוימות, מה שהופך את הכלי רגרסיה לוגיסטית לבחירה טובה.

23) **k-Nearest Neighbors (k-NN)** – הביצועים הגרועים ביותר, ככל הנראה משום שהמרחקים מאבדים מהמשמעות שלהם במימדים גבוהים.

תוצאות:

בהרצה הראשונה שלי השתמשתי ב- 19 תכונות רלוונטיות מהמאגר וקיבלתי את התוצאות הבאות:

```
Filtered young players (FIFA 2021 and below, first appearance only): 9138 rows
Remaining rows after defining target: 3639

Model Evaluation Results:
                    Model  Mean Absolute Error  Root Mean Squared Error
0           Decision Tree             6.073003                 7.562868
1  Support Vector Machine             4.539945                 5.706311
2                AdaBoost             4.527548                 5.704017
3     Logistic Regression             4.771350                 6.113481
4        Nearest Neighbor             6.203857                 7.750589
```

ה- Mean Absolute Error הטוב ביותר מגיע מ- AdaBoost ו- SVM.

הכוונה היא כי בממוצע, החיזויים שגויים בכ- 4.5 נקודות דירוג כללי.

התוצאות לא גרועות אך הן גם לא מעולות. (מעולה משמעותו $MAE \leq 3$)

ה- Root Mean Squared Error הטוב ביותר מגיע גם הוא מ- AdaBoost ו- SVM.

RMSE נמוך יותר פירושו פחות שגיאות גדולות, אך ערך של 5.7 עדיין מצביע על כך שהמודל עושה כמה טעויות משמעותיות.

על מנת לשפר את התוצאות נעזרתי בכלים מסוימים:

- **feature selection:**

ייתכן שחלק מהתכונות פחות חשובות, בעוד שאחרים עשויים לעזור באופן חזק יותר בחיזוי.
שיטה שיכולה לעזור בלסנן את התכונות הפחות חשובות היא SHAP analysis.
SHAP מקצה לכל תכונה ערך המייצג כמה היא תורמת לחיזוי מסוים.
היא מחשבת את תרומת כל תכונה על ידי חישוב התרומה השולית הממוצעת שלה על פני כל תתי הקבוצות האפשריות של התכונות.

```
def feature_selection(X_train, y_train, features):
    rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
    rf_model.fit(X_train, y_train)
    feature_importances = rf_model.feature_importances_
    feature_importance_df = pd.DataFrame({'Feature': features, 'Importance': feature_importances})
    feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)
    print("Feature Importance Analysis:")
    print(feature_importance_df)
    return feature_importance_df
```

המימוש של השיטה נעשה על ידי שימוש ב- Random Forest – כלי למידת מכונה שמתאימה מספר עצי החלטה על מנת לחזות ערכים רציפים.
feature_importances_ היא מתודה של ה- Random Forest שאומן על נתוני האימון ומספקת את החשיבות של תכונה בחיזוי ה- y_train.
לאחר הרצה של feature_selection קיבלתי רשימה של כל התכונות והחשיבות שלהן בסדר יורד ומתוך הרשימה השארתי רק את 10 התכונות בעלי החשיבות הגדולה ביותר.

```
Filtered young players (FIFA 2021 and below, first appearance only): 9138 rows
Remaining rows after defining target: 3639
Feature Importance Analysis:
                      Feature  Importance
0                   potential    0.119038
18                   wage_eur    0.114765
17                  value_eur    0.088737
2          skill_ball_control    0.067854
3     attacking_short_passing    0.061122
13  defending_standing_tackle    0.057772
12              power_stamina    0.053736
11             power_strength    0.052293
4          attacking_crossing    0.045338
14   defending_sliding_tackle    0.045164
7            power_shot_power    0.043284
5            mentality_vision    0.040919
9        movement_acceleration    0.040447
10       movement_sprint_speed    0.039611
8            power_long_shots    0.039145
6          attacking_finishing    0.036928
1                   dribbling    0.035130
16                  weak_foot    0.010999
15                skill_moves    0.007718
```

- **Hyperparameter Tuning:**

מציאה של פרמטרים טובים יותר לכל מודל.

```python
def hyperparameter_tuning(X_train, y_train):
    param_grid = {
        'C': [0.1, 1, 10, 100],
        'kernel': ['linear', 'rbf']
    }
    grid_search = GridSearchCV(SVC(), param_grid, cv=5, scoring='accuracy')
    grid_search.fit(X_train, y_train)
    print(f"Best parameters for SVC: {grid_search.best_params_}")
    return grid_search.best_estimator_
```

זאת פונקציה שמצאתי בגוגל למציאה של פרמטרים טובים יותר ב- SVM.
הפונקציה הזאת הופעלה גם על שאר המודלים.

התוצאות שהתקבלו לגבי היפרפרמטריים יעילים יותר:

```
C:\Users\yuval\Desktop\Code\PyCharm Code Projects\FinalProjectML\.venv\Lib\si
less than n_splits=3.
  warnings.warn(
Best parameters for SVC: {'C': 0.1, 'kernel': 'linear'}
C:\Users\yuval\Desktop\Code\PyCharm Code Projects\FinalProjectML\.venv\Lib\si
less than n_splits=3.
  warnings.warn(
Best parameters for AdaBoost: {'learning_rate': 1, 'n_estimators': 100}
C:\Users\yuval\Desktop\Code\PyCharm Code Projects\FinalProjectML\.venv\Lib\si
less than n_splits=3.
  warnings.warn(
Best parameters for RandomForest: {'max_depth': 20, 'n_estimators': 100}
C:\Users\yuval\Desktop\Code\PyCharm Code Projects\FinalProjectML\.venv\Lib\si
less than n_splits=3.
  warnings.warn(
Best parameters for LogisticRegression: {'C': 0.1}
C:\Users\yuval\Desktop\Code\PyCharm Code Projects\FinalProjectML\.venv\Lib\si
less than n_splits=3.
  warnings.warn(
Best parameters for KNN: {'n_neighbors': 7, 'weights': 'uniform'}
Total runtime: 25.82 seconds
(.venv) PS C:\Users\yuval\Desktop\Code\PyCharm Code Projects\FinalProjectML>
```

לאחר מכן ביצעתי שינוי ידני בהיפרפרמטרים של k-NN ו- AdaBoost כי מצאתי ירידה קלה ב- MAE.

ב- k-NN: ‘n_neighbors’ : 10

ב- AdaBoost: ‘n_estimators’: 300 , ‘learning_rate’: 0.1

קיבלתי את התוצאות הבאות:

```
Filtered young players (FIFA 2021 and below, first appearance only): 9138 rows
Remaining rows after defining target: 3639

Model Evaluation Results:
                    Model  Mean Absolute Error  Root Mean Squared Error
0           Decision Tree             6.002755                 7.550564
1  Support Vector Machine             4.436639                 5.706673
2                AdaBoost             4.314050                 5.549675
3     Logistic Regression             4.586777                 5.809416
4        Nearest Neighbor             5.524793                 7.014841
Total runtime: 2.33 seconds
```

ישנה ירידה קלה ב-MAE אך לא כמצופה.

ה- MAE לא ירד מ-3 ולכן התוצאות אינן מעולות אך הן די טובות.

מסקנות:

17) AdaBoost הוא המודל הטוב ביותר לשאלה הזאת.
זה תואם ליכולת של AdaBoost לשלב מספר "weak learners" כדי לשפר את הביצועים הכוללים.

18) SVM הוא המודל השני הטוב ביותר לשאלה הזאת, מה שמעיד על כך שהקשר בין התכונות לליבלים עושי להילכד היטב באמצעות גישה המבוססת על היפר-מישור.

19) Decision Tree עם תוצאות יותר גרועות מה שמעיד על כך שעץ החלטה בודד כנראה יוצר overfitting ומתקשה בהכללה לנתונים חדשים.
גם k-NN עם תוצאות גרועות, מה שמעיד על כך שהמודל מתקשה בגלל שהוא לא מצליח למצוא באופן יעיל קשרים לא ליניאריים בנתונים.

20) AdaBoost משיג ביצועים טובים יותר משמעותית מעץ החלטה יחיד, מה שמראה כי Boosting משפר את דיוק החיזויים.

21) כל המודלים הפיקו תועלת מנרמול התכונות על ידי StandardScaler, מה שככל הנראה שיפר את הביצועים של SVM ו- AdaBoost.

22) התכונות שנבחרו לאחר סינון נראות כמכריעות בתהליך חיזוי הדירוג הכללי העתידי של שחקן במשחק.

אתגרים:

13)**הפחתה של MAE:**

הרעיון היה להפחית את MAE להיות פחות מ-3 וגם לאחר feature selection ו-hyperparameter tuning לא הצלחתי להפחית את MAE יותר מדי.

טכניקות:

- עבדו טוב: Logistic Regression ,SVM ,Adaboost.

AdaBoost עבד טוב יותר מפני שהוא משלב מספר "weak learners" ובכך מפחית bias ושונות. כמו כן, טכניקות של Boosting עוזרות להתמקד במקרים שקשה לחזות.
SVM ככל הנראה מצא היפר-מישור טוב שמפריד את ה-data.
Logistic Regression ככל הנראה הצליח לזהות קשרים ליניאריים שימושיים בין התכונות לדירוגים העתידיים של השחקנים.

- עבדו פחות טוב: Decision Trees, k-NN

k-NN עבד פחות טוב ככל הנראה בגלל שהתכונות של שחקני FIFA נמצאות בטווחים שונים, מה שמקשה על שיטות שמבוססות על מרחק לבצע הכללה.
Decision Tree עבד פחות טוב ככל הנראה בגלל שעצי החלטה נוטים לשנן את נתוני האימון ולגרום ל- overfitting.

שאלה 6:

מהם המאפיינים החשובים ביותר שקובעים את הדירוג הכולל של השחקן (overall)?

קישור לקוד המתאים: https://github.com/Yuval10Dahan/Final-Project-ML/tree/main/Q6

כלים שהרצתי:

24) **Random Forest** – מאפשר מדידת חשיבות של תכונות על ידי בדיקה כמה כל תכונה תורמת להפחתת השגיאה בחיזוי.
מתמודד היטב עם נתונים במימד גבוה.

25) **PCA** – טכניקה להפחתת מימד.
שימושית עבור הסרת תכונות מיותרות תוך שמירה על הנתונים המשמעותיים ביותר.
בשאלה הזאת בוצעה סטנדרטיזציה לנתונים כדי להבטיח שכל התכונות יקבלו משקל שווה.

תוצאות:

התהליך מחולק ל-3 שלבים.

שלב ראשון:

נעשה סינון כך שהתכונות היחידות שמשתתפות הן תכונות בעלות ערך מספרי.

נעשית השמטה של התכונה "overall" כי זהו הלייבל שלנו.

נעשה אימון ל-Random Forest על ה-training data.

נעשה חילוץ של חשיבות כל תכונה לחיזוי לפי סדר יורד.

לאחר מכן מתבצעת סטנדרטיזציה (מתן משקל שווה לכל תכונה) והפעלת PCA להורדת מימד תוך שמירה על הנתונים המשמעותיים.

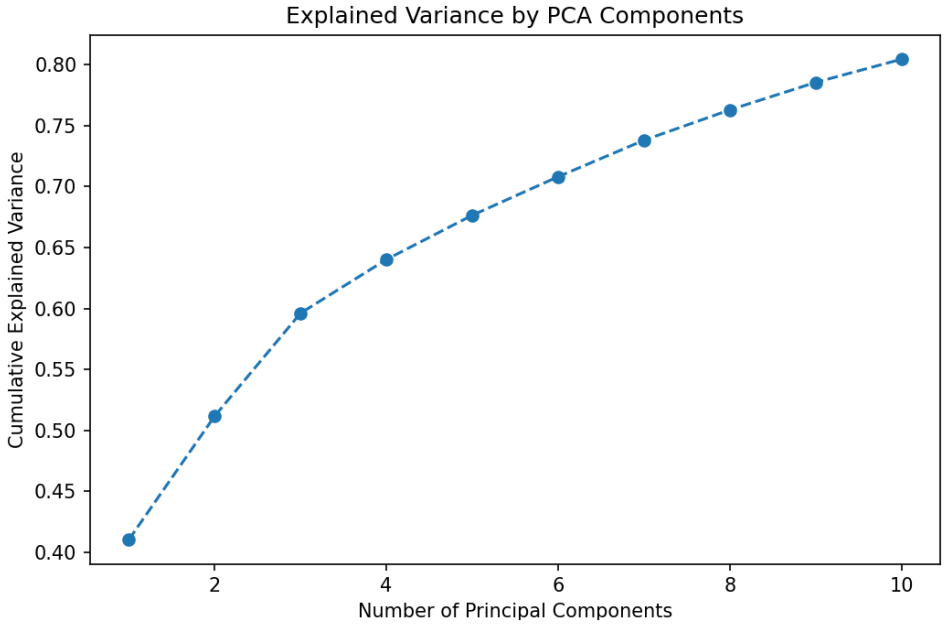
לאחר מכן מוצגים 10 התכונות החשובות ביותר לקביעת דירוג כולל של שחקן.

```
Top 10 Important Features:
                      Feature  Importance
4                   value_eur    0.720125
40         movement_reactions    0.144927
6                         age    0.037119
3                   potential    0.036575
0                   player_id    0.032361
1                fifa_version    0.012714
5                    wage_eur    0.003166
19  international_reputation    0.002062
25                  defending    0.001656
9                club_team_id    0.000776
```

ניתוח נתונים:

1) שווי שוק החשוב ביותר – דירוגי FIFA בדרך כלל משקפים את ערך השוק של השחקן, שכן שחקנים בכירים נוטים להיות יקרים יותר.
2) מהירות תגובה – דירוגי FIFA נוטים להעדיף שחקנים עם מהירות תגובה גבוהה, שכן היא משקפת זריזות בזמן משחק (מי שמשחק במשחק יודע להגיד שזה חשוב מאוד).
3) גיל ופוטנציאל משפיעים במידה בינונית – שחקנים צעירים עם פוטנציאל גבוה יכולים להגיע לדירוגים גבוהים (יכולים גם שלא אבל לרוב כן יגיעו).
4) שכר, מוניטין בינלאומי ויכולות הגנה משפיעים בצורה די חלשה – השכר לא תמיד תואם לדירוג הכללי (חלק מהשחקנים מרוויחים יותר מדי או פחות מדי ביחס ליכולת שהם מציגים). מוניטין יכול לעזור אך הוא לא גורם ישיר בחישוב הדירוג הכללי (ישנם שחקנים שלא מציגים יכולת טובה בנבחרת אך בקבוצה שלהם הם מעולים, ולהפך).
5) המועדון של השחקן כמעט חסר תועלת – לא קובע ישירות דירוג כולל של שחקן, ישנם שחקנים מצוינים שמשחקים בקבוצות קטנות יותר.

נוצר לנו גם גרף המציג את השונות אל מול מספר הרכיבים של ה-PCA:



ניתוח הגרף:

1) 10 הרכיבים הראשונים מסבירים כ-80% מהשונות, כלומר ניתן להסיר חלק מהתכונות הפחות חשובות.
2) שווי שוק ומהירות תגובה כנראה שולטים ב-2 הרכיבים הראשונים – מכיוון שתכונות אלו נמצאות בקורלציה גבוהה עם הדירוג הכולל, הן כנראה נתפסות על ידי הרכיב הראשון והשני.
3) בגלל ש-5 הרכיבים הראשונים מסבירים כ-70% מהשונות, רוב המידע במאגר הנתונים מרוכז ברכיבים הללו. השונות הנותרת מפוזרת על פני רכיבים חלשים רבים, מה שאומר שחלק מהתכונות תורמות מעט מאוד מידע שהוא ייחודי וחשוב.
משמע, מודל שמשתמש רק ב-5 הרכיבים הראשונים או אפילו קצת יותר עדיין עשוי לספק תוצאות טובות.

שלב שני:

על מנת לשפר את התוצאות, הוסרו 2 תכונות שמשפיעות לרעה על החיזוי ונחשבות כ- bias.

התכונות הללו הן: player_id , fifa_version .

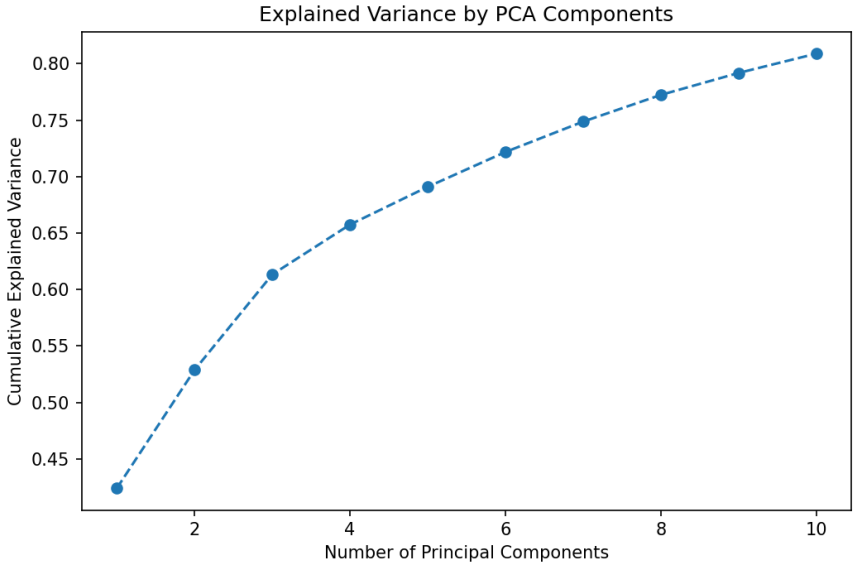
```
Top 10 Important Features (After Bias Removal):
                          Feature  Importance
2                       value_eur    0.713711
38             movement_reactions    0.153721
4                             age    0.047998
1                       potential    0.037527
11  club_contract_valid_until_year    0.013466
3                        wage_eur    0.008380
50            mentality_composure    0.002869
23                      defending    0.002830
17        international_reputation    0.002485
18              release_clause_eur    0.001233
```

נוצר שינוי ברשימה של 10 התכונות החשובות ביותר.

ניתוח נתונים:

1) ירידה מזערית בחשיבות של שווי שוק מעידה על כך שהסרה של תכונות שהן bias נתנה לתכונות אחרות להיות רלוונטיות יותר.
2) מהירות תגובה הפכה לחשובה יותר.
3) גיל ופוטנציאל הפכו לחשובות יותר.
4) התווספו תכונות חדשות:
   - אורך חוזה של שחקן – מועדונים מאריכים חוזים לשחקנים טובים ממש.
   - מנטליות – מראה כי שחקנים עם חוסן מנטלי נוטים לקבל דירוג גבוה יותר.
   - שכר – הפך לרלוונטי יותר, כלומר שכר נמצא בקורלציה עם דירוג כולל, אך הרבה פחות משווי שוק.
   - הגנה – כעת בעל חשיבות גדולה יותר.

הסרת תכונות bias שיפרה את יכולות הפרשנות של המודל.



אין כל כך הבדל בהשפעה על הגרף והנתונים די דומים לגרף הקודם.

משמע, ניתוח ה- PCA עדיין מצביע על כך שהרבה מהתכונות הן מיותרות וניתן לצמצם עוד יותר את המימדיות. הגורמים המרכזיים המשפיעים על השונות הם עדיין שווי שוק, מהירות תגובה, גיל ופוטנציאל.

שלב שלישי ואחרון:

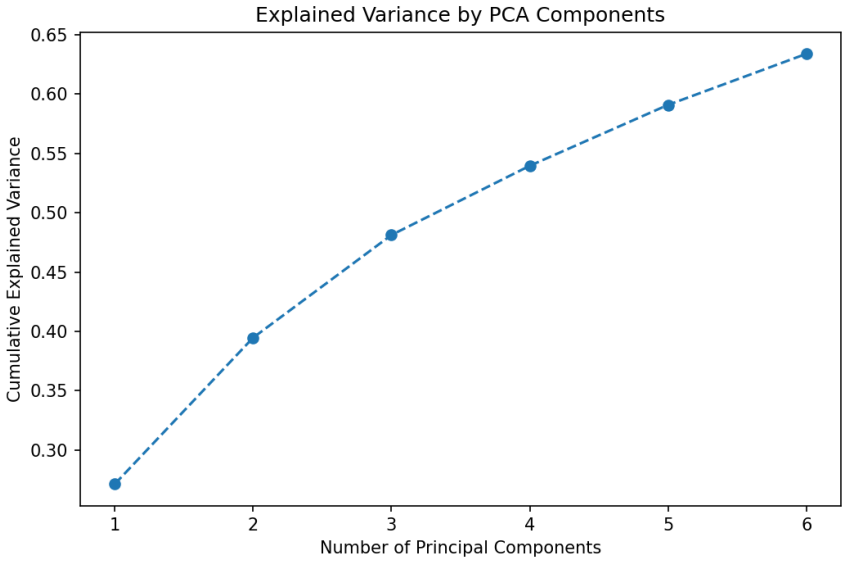
על מנת לשפר עוד יותר את התוצאות, נוספה פונקציה חדשה שמסירה תכונות עם קורלציה מעל 0.85 כדי להימנע ממידע מיותר. זה מבטיח שהמודלים לא יוטעו על ידי תכונות שתלויות זו בזו ברמה גבוהה מאוד.

כמו כן, מספר הרכיבים ב-PCA שונה ל-6 במקום 10.

```
Top 10 Important Features (After Refinement):
                          Feature  Importance
2                       value_eur    0.714281
24              movement_reactions    0.154029
4                             age    0.048452
1                       potential    0.038307
11  club_contract_valid_until_year    0.013822
3                        wage_eur    0.008655
20                      defending    0.003742
32             mentality_composure    0.003359
17        international_reputation    0.002494
7                     club_team_id    0.001372
```

ניתוח נתונים:

1) שווי שוק, מהירות תגובה, גיל ופוטנציאל נותרו התכונות החשובות ביותר. הסרת תכונות עם קורלציה גבוהה לא שינתה את הדומיננטיות שלהן, מה שמאשר את חשיבותן לדירוג הכללי.
2) הגנה ומנטליות הפכו לחשובים יותר – יכול להעיד על כך ש-FIFA מדרגת שחקני הגנה באופן שונה משחקני התקפה ומיומנות הגנתית תורמת באופן ייחודי לדירוג הכללי.
3) המועדון של השחקן חזר לרשימה אך עם חשיבות נמוכה – כנראה יש לו רלוונטיות מסוימת אך לא משמעותי לחישוב הדירוג הכולל. כמו כן, יכול להיות שזאת תכונה bias.



ניתוח הגרף:

1) רק 6 רכיבים מסבירים את רוב השונות (כ-65%).
זה מצביע על כך שהרבה תכונות היו מיותרות ולא הוסיפו מידע משמעותי חדש.

2) השונות הכוללת ירדה מעט – הירידה מצביעה על כך שחלק מהמידע השימושי אבד לאחר הסרת התכונות בעלות הקורלציה הגבוהה.
עם זאת, בגלל שהרכיבים של PCA עדיין מסבירים חלק משמעותי מהשונות (כ-65%), הנתונים עדיין נותנים מידע רב.
3) מכיוון שרוב השונות נתפסת על ידי 6 רכיבים, ניתן ככל הנראה לאמן מודל יעיל באמצעות התכונות הללו בלבד.

מסקנות:

23) לפי ניתוח PCA אפשר להגיד שגם תת קבוצה של תכונות מסך התכונות הנבדקות, יכולה להסביר את רוב השונות ב-data.
24) הסרה של תכונות עם קורלציה גבוהה מונעת מידע מיותר בחיזוי.

אתגרים:

14) **תכונות bias** – בהתחלה התכונות "player_id", "fifa_version" הופיעו בין התכונות החשובות. תכונות אלו אינן משקפות מיומנות של שחקן, לכן שמירה על תכונות אלו הייתה גורמת לציוני חשיבות גבוהים שהם מלאכותיים.
הפתרון היה להסיר את התכונות הללו.
15) **קורלציה גבוהה בין תכונות** – סעיף שחרור בעל קורלציה גבוהה עם שווי שוק ושכר. שמירה על כל שלושת התכונות הללו מובילה לעודף מידע, מה שגורם למודל להתמודד יתר על המידה במדדים פיננסיים.
הפתרון היה להסיר תכונות עם קורלצייה גבוהה מעל 0.85.
16) בכל שלב תכונות חדשות התגלו כחשובות יותר. הפתרון הוא להסיר תכונות bias ולבצע מספר איטרציות של תהליך בחירת התכונות החשובות ביותר, כדי לוודא שהתוצאות עקביות.

טכניקות:

- עבדו טוב: Random Forest, PCA

<u>שאלות נוספות:</u>

היו לי עוד 2 שאלות שרציתי לבחון אך התוצאות היו די גרועות (accuracy < 65) ולצערי לא היה לי זמן לנסות לשפר את התוצאות.

להלן השאלות:

1) האם ליגות שונות נותנות עדיפות למאפיינים שונים? (למשל, דגש על פיזיות בליגה האנגלית לעומת יכולת טכנית בליגה הספרדית).
2) מצא אילו שחקנים בעלי סיבולת גבוהה ועמידות לפציעות, מתאימים לשחק בליגות אינטנסיביות (למשל, ליגה אנגלית).