



עבודה במדעי הנתונים

איתמר בר און, שחר בר אור ויהונתן ריאהי

[קישור לקולאב](#)

https://colab.research.google.com/drive/1wwVFE_5j0yp6_Ak-xdrJKcUiLw34pdno?usp=sharing



מבוא ומטרת המשחק

בלאקג'ק הוא משחק קלפים פופולרי שנמצא בקזינו ברחבי העולם.

המשחק נערך בין השחקן לבין הדילר, והמטרה היא להגיע לסכום קלפים של 21 או קרוב לכך, בלי לחרוג ממנו.

אם השחקן עובר 21, הוא מפסיד אוטומטית.

השחקן מתחרה מול הדילר, ולא מול שחקנים אחרים.



שלבי המשחק ניצחון והפסד

1. **הימור:** השחקן מציב סכום כסף לפני תחילת המשחק.
2. **קבלת קלפים:** השחקן והדילר מקבלים שני קלפים.
3. **החלטות השחקן:** השחקן יכול לבחור:
 - "סטנד" – לשמור על הקלפים הנוכחיים.
 - "היט" – למשוך קלף נוסף.
4. **תור הדילר:** הדילר חייב למשוך קלפים עד שהסכום שלו מגיע ל-17 ומעלה.
 - אם השחקן קרוב ל-21 מבלי לעבור, והוא גבוה יותר מהדילר, הוא מנצח.
 - אם הדילר עובר 21, השחקן מנצח אוטומטית.
 - אם שני הצדדים הגיעו לאותו סכום, זה תיקו.



מספר השורות, העמודות וסוגי העמודות
הקובץ מכיל 900,000 שורות ו-21 עמודות.

עמודות מספריות:

- קלפים: (*card1, card2, card3, card4, card5, dealcard1, dealcard2, dealcard3,*
dealcard4, dealcard5)
 - טווח ערכים: 0-11 (0 מציין שאין קלף נוסף)
 - סוג משתנה: בדיד
- סכומי הקלפים: (*sumofcards, sumofdeal, ply2cardsum*)
 - *sumofcards, sumofdeal*: טווח ערכים 8-26
 - *ply2cardsum*: טווח ערכים 2-21
 - סוג משתנה: בדיד
- סכומי זכיות: (*plwinamt, dlwinamt*)
 - טווח ערכים: 0-25
 - סוג משתנה: בדיד



המשך

עמודות קטגוריות:

- תוצאת המשחק: (*winloss*)
 - ערכים אפשריים: Win/Loss
- בלקג'ק: (*blkjck*)
 - ערכים אפשריים: win/nowin
- סטטוסים: (*plybustbeat, dlbustbeat*)
 - מציינים אם השחקן או הדילר הפסידו בעקבות חריגה מ-21.

קשרים בין עמודות

- הסכום הכולל של הקלפים (sumofcards) משפיע ישירות על הזכייה או ההפסד (winloss).
- הסכום הכולל של קלפי הדילר (sumofdealer) משפיע על הסיכויים של השחקן לנצח.
- אם plybustbeat מופיע, השחקן חרג מ-21 והפסיד אוטומטית.
- אם dlbustbeat מופיע, הדילר חרג מ-21 והשחקן מנצח אוטומטית.
- אם השחקן קיבל 21 בשני קלפים (ply2cardsum = 21), זה מוביל ל-Blackjack (blkjck = win) וכנראה גם לניצחון (אם הדילר לא הוציא גם BlackJack).

מציאת משתני NaN

בטבלא לא אמורים להיות משתני NaN אך מדי פעם יש בעיות בהעלאה של הקובץ ואז יש משתני NaN.

למשל פה שיש שורה אחת שבה יש משתני NaN שאנחנו מורידים.

```
# search every row and column and show only the NaN values
nanFind = df[df.isna().any(axis=1)]
nanFind
```

	Unnamed: 0	PlayerNo	card1	card2	card3	card4	card5	sumofcards	dealcard1	dealcard2	...	dealcard4	dealcard5	sumofdeal	blkjck	winloss	plybustbeat	dlbustbeat	plwinamt	dlwinamt	ply2cardsum
320838	0	Player1	10	10	0	0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

1 rows x 21 columns

יצירת משתנים

פה אנחנו הוספנו 2 משתנים שמראים כמה קלפים לקח השחקן וכמה לקח הדילר.

עשינו זאת בכך שעברנו בכל שורה מה הקלף האחרון שכל אחד לקח ויצרנו לזה רשומה מכיוון שלפני פשוט היה רק איזה קלפים הם קיבלו וכשהם הפסיקו לקחת ברשומות היה 0.

אז כשראינו 0 הפסקנו לספור כמה קלפים הם לקחו ורשמנו

```
[ ] def find_hands(row):  
    # get the data of the current row from df  
    curr_row = ndf.iloc[row]  
    last_player_card = 1  
    last_dealer_card = 1  
  
    while last_player_card < 6 and curr_row[f"card{last_player_card}"] != 0:  
        last_player_card += 1  
    while last_dealer_card < 6 and curr_row[f"dealcard{last_dealer_card}"] != 0:  
        last_dealer_card += 1  
    return last_player_card - 1, last_dealer_card - 1
```

```
[ ] def create_counting():  
    #add to ndf the ammount of cards that the player played and the ammount of cards that the dealer played  
    for i in range(len(ndf)):  
        last_player_card, last_dealer_card = find_hands(i)  
        ndf.at[i, 'last_player_card'] = last_player_card  
        ndf.at[i, 'last_dealer_card'] = last_dealer_card  
  
    create_counting()
```


יצירת משתנים

```
[ ] def get_bust_chance(row):  
    """  
    take the final hand score of the player and the cards that the player got  
    calculate what were the chances that if he had taken another card they would bust  
    """  
  
    current_row = ndf.iloc[row]  
    player_hand = current_row['sumofcards']  
  
    #calculate how many possible cards we can get to bust  
    smallest_card_to_bust = 21 - player_hand  
    possible_busts = (13 - smallest_card_to_bust) * 4  
    for card in range(1, int(current_row["last_player_card"]+1)):  
        if current_row[f"card{card}"] >= smallest_card_to_bust:  
            possible_busts -= 1  
    return possible_busts / 52
```

```
def set_bust_category():  
    for i in range(len(ndf)):  
        chances = get_bust_chance(i)  
        ndf.at[i, 'bust_chance'] = chances  
    set_bust_category()
```

יצרנו את העמודה הזו כדי לקבל מה הסיכויים ששחקן עשה STAND לפני מתי שהיה אמור לעשות ומה היו הסיכויים שלו לעשות BUST באותו רגע. הסיבה שהסיכויים יכולים להיות גם יותר גדולים מאחד זה מכיוון שאז זה אומר שהשחקן כבר עבר את ה21 אך הם לא 1 כי הוא לא על 21 ואנחנו גם רוצים לראות נתונים של אנשים שעברו את ה21.

יצירת משתנים

יצרנו את diff in taking כדי לבדוק אם יש להבדל בין הכמות שהשחקן לקח לדילר השפעה על התוצאות של המשחק.

יצרנו את העמודה בכך שחילקנו בין כמות הקלפים שהשחקן לקח לכמות שהדילר לקח.

```
[ ] def diff_in_taking(row):  
    """  
    take the final hand score of the player and the cards that the player got  
    calculate the difference between the player hand to the dealer hand  
    """  
  
    current_row = ndf.iloc[row]  
    player_hand = current_row['last_player_card']  
    dealer_hand = current_row['last_dealer_card']  
  
    return (float(player_hand) / dealer_hand)
```

```
[ ] def set_diff_in_taking():  
    for i in range(len(ndf)):  
        diff = diff_in_taking(i)  
        ndf.at[i, 'diff_in_taking'] = diff  
    set_diff_in_taking()
```

```
[ ] ndf
```

יצירת משתנים

יצרנו את DIFFERENCE כדי לבדוק איך ההבדל בין היד של השחקן לשל הדילר משפיעה על הנתונים שבמשחק.

עשינו זאת בכך שחילקנו בין סכום הידיים שלו

```
[ ] def get_difference(row):  
    """  
    take the final hand score of the player and the cards that the player got  
    calculate the difference between the player hand to the dealer hand  
    """  
    current_row = ndf.iloc[row]  
  
    player_hand = current_row['sumofcards']  
    dealer_hand = current_row['sumofdeal']  
  
    return (float(player_hand) / dealer_hand)
```

```
[ ] def set_difference():  
    for i in range(len(ndf)):  
        difference = get_difference(i)  
        ndf.at[i, 'difference'] = difference  
    set_difference()
```

טעינת data frame

	Unnamed: 0	PlayerNo	card1	card2	card3	card4	card5	sumofcards	dealcard1	dealcard2	...	dealcard4	dealcard5	sumofdeal	blkjck	winloss	plybustbeat	dlbustbeat	plwinamt	dlwinamt	ply2cardsum
0	0	Player1	7	10	0	0	0	17	10	8	...	0	0	18	nowin	Loss	Beat	Dlwin	0	10	17
1	1	Player2	10	9	0	0	0	19	10	8	...	0	0	18	nowin	Win	Plwin	Beat	20	0	19
2	2	Player3	9	8	0	0	0	17	10	8	...	0	0	18	nowin	Loss	Beat	Dlwin	0	10	17
3	3	Player4	2	10	0	5	0	17	10	8	...	0	0	18	nowin	Loss	Beat	Dlwin	0	10	12
4	4	Player5	10	2	0	5	0	17	10	8	...	0	0	18	nowin	Loss	Beat	Dlwin	0	10	12
...
899995	1	Player2	10	7	0	0	0	17	3	9	...	2	0	18	nowin	Loss	Beat	Dlwin	0	10	17
899996	2	Player3	6	1	10	0	0	17	3	9	...	2	0	18	nowin	Loss	Beat	Dlwin	0	10	7
899997	3	Player4	4	2	9	0	0	15	3	9	...	2	0	18	nowin	Loss	Beat	Dlwin	0	10	6
899998	4	Player5	9	10	0	0	0	19	3	9	...	2	0	18	nowin	Win	Plwin	Beat	20	0	19
899999	5	Player6	5	10	0	0	0	15	3	9	...	2	0	18	nowin	Loss	Beat	Dlwin	0	10	15

900000 rows x 21 columns

קישור: <https://www.kaggle.com/datasets/mojocolors/900000-hands-of-blackjack-results>

מידע על הdata frame



```
df.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 900000 entries, 0 to 899999
Data columns (total 21 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Unnamed: 0      900000 non-null  int64
1   PlayerNo        900000 non-null  object
2   card1           900000 non-null  int64
3   card2           900000 non-null  int64
4   card3           900000 non-null  int64
5   card4           900000 non-null  int64
6   card5           900000 non-null  int64
7   sumofcards      900000 non-null  int64
8   dealcard1       900000 non-null  int64
9   dealcard2       900000 non-null  int64
10  dealcard3       900000 non-null  int64
11  dealcard4       900000 non-null  int64
12  dealcard5       900000 non-null  int64
13  sumofdeal       900000 non-null  int64
14  blkjck          900000 non-null  object
15  winloss         900000 non-null  object
16  plybustbeat     900000 non-null  object
17  dlbustbeat      900000 non-null  object
18  plwinamt        900000 non-null  int64
19  dlwinamt        900000 non-null  int64
20  ply2cardsum     900000 non-null  int64
dtypes: int64(16), object(5)
memory usage: 144.2+ MB
```

פעולות Group by על משתנים

```
#GroupBy
df['winloss'] = df['winloss'].replace('Push', 'Draw')

groupby_columns = ['winloss', 'sumofcards', 'blkjck']
agg_columns = ['sumofdeal']

# Calculate mean
mean_df = df.groupby(groupby_columns)[agg_columns].mean()

# Calculate median
median_df = df.groupby(groupby_columns)[agg_columns].median()

# Display results
print_section(["Mean", mean_df])
print_section(["Median", median_df])
```

Median				Mean			
winloss	sumofcards	blkjck	sumofdeal	winloss	sumofcards	blkjck	sumofdeal
Draw	12	nowin	12.0	Draw	12	nowin	12.000000
	13	nowin	13.0		13	nowin	13.000000
	14	nowin	14.0		14	nowin	14.000000
	15	nowin	15.0		15	nowin	15.000000
	16	nowin	16.0		16	nowin	16.000000
	17	nowin	17.0		17	nowin	17.000000
	18	nowin	18.0		18	nowin	18.000000
	19	nowin	19.0		19	nowin	19.000000
	20	nowin	20.0		20	nowin	20.000000
	21	Win	21.0		21	Win	21.000000
Loss		nowin	21.0	Loss		nowin	21.000000
	9	nowin	18.0		9	nowin	18.000000
	10	nowin	18.5		10	nowin	18.666667
	11	nowin	19.0		11	nowin	18.785714
	12	nowin	19.0		12	nowin	18.854284
	13	nowin	19.0		13	nowin	18.822057
	14	nowin	19.0		14	nowin	18.808846
	15	nowin	19.0		15	nowin	18.849632
	16	nowin	19.0		16	nowin	18.855194
	17	nowin	20.0		17	nowin	19.512758
Win	18	nowin	20.0		18	nowin	19.978039
	19	nowin	20.0		19	nowin	20.388277
	20	nowin	21.0		20	nowin	21.000000
	22	nowin	20.0		22	nowin	20.101145
	23	nowin	20.0		23	nowin	20.070077
	24	nowin	20.0		24	nowin	20.111371
	25	nowin	20.0		25	nowin	20.077034
	26	nowin	20.0		26	nowin	19.974144
	9	nowin	25.0		9	nowin	25.000000
	10	nowin	24.0		10	nowin	24.000000
	11	nowin	24.0	Win	11	nowin	23.777778
	12	nowin	24.0		12	nowin	23.823289
	13	nowin	23.0		13	nowin	23.569858
	14	nowin	23.0		14	nowin	23.555628
	15	nowin	24.0		15	nowin	23.583784
	16	nowin	24.0		16	nowin	23.563986
	17	nowin	24.0		17	nowin	23.646390
	18	nowin	22.0		18	nowin	21.280105
	19	nowin	18.0		19	nowin	20.448244
	20	nowin	19.0		20	nowin	20.214450
	21	Win	20.0		21	Win	20.216725
		nowin	20.0			nowin	20.074685

מסקנות מ-Group by

1. יחסי סכומי קלפים בין השחקן לדילר

כאשר השחקן מפסיד:

- כאשר לשחקנים יש יד חלשה (8-16), לדילר יש בממוצע יד של 18-19
- עבור ידי שחקן של 17-20, הציון הממוצע של הדילר עולה (19.5-21), מה שמסביר את ההפסד
- שחקנים עם סכום קלפים של 22+ (ידיים שחרגו) מפסידים לדילרים עם יד ממוצעת של ~20, המצביע על כך שהדילר לא נדרש להסתכן חריגה

כאשר השחקן מנצח:

- עבור ידיים נמוכות של השחקן (9-17), הסכום הממוצע של הדילר גבוה מאוד (23-24), המצביע על חריגה הדילר
- עבור ידיים חזקות יותר של השחקן (18-20), היד הממוצעת של הדילר יורדת מה שמרמז שהשחקן ניצח על-ידי יד בעלת ערך טוב יותר
- עבור שחקנים עם 21 נקודות, הממוצע של הדילר הוא בערך 20.

במקרה של תיקו:

- החציון של סכום הדילר תואם בדיוק את סכום הקלפים של השחקן, מה שמאשר שאלה תיקו אמיתי

2. בלק'ק לעומת ניצחונות רגילים

- כאשר לשחקנים יש 21 נקודות, יש הבחנה בין ניצחונות בלק'ק ("Win") לבין 21 רגיל ("nowin")
- לשני הסוגים יש ממוצעי יד דילר דומים (~20.2)

3. השלכות אסטרטגיות

- להישאר עם 17-20: הנתונים תומכים באסטרטגיית בלק'ק בסיסית של עמידה על ערכים אלה, שכן הדילר לעיתים קרובות חורג כאשר שחקנים מנצחים עם סכומים אלה
- ידיים נמוכות (8-11): שחקנים עם סכומים אלה מנצחים אך ורק כאשר הדילרים חורגים עם סכומים מעל 21 (מוצג על ידי יד דילר ממוצעת ~23-24)
- תדירות תיקו: תיקו מתרחש בעקביות כאשר לשחקן ולדילר יש אותו סכום בדיוק.

פעולות value count על משתנים

```
#value counts
# Perform Value Counts
value_counts_winloss = df["winloss"].value_counts()
value_counts_blkjck = df["blkjck"].value_counts()
value_counts_sumofcards = df["sumofcards"].value_counts()

# Function to print sections neatly
def print_section(title, data):
    print("\n" + "=" * 50)
    print(f"{title}")
    print("=" * 50)
    print(data)
    print("=" * 50 + "\n")

print_section("Value Counts - Player's Win/Loss", value_counts_winloss)
print_section("Value Counts - Player's Blackjack Status", value_counts_blkjck)
print_section("Value Counts - Player's Sum of Cards", value_counts_sumofcards)
```

Value Counts - Player's Win/Loss

```
winloss
Loss    152757
Win     138044
Push     30037
Name: count, dtype: int64
```

Value Counts - Player's Blackjack Status

```
blkjck
nowin   305483
Win      15355
Name: count, dtype: int64
```

Value Counts - Player's Sum of Cards

```
sumofcards
20.0    52386
18.0    39728
17.0    38321
19.0    37662
21.0    33586
22.0    16099
13.0    13714
14.0    13414
15.0    13102
23.0    12887
16.0    12843
24.0    11238
25.0     9668
12.0     8384
26.0     7755
```



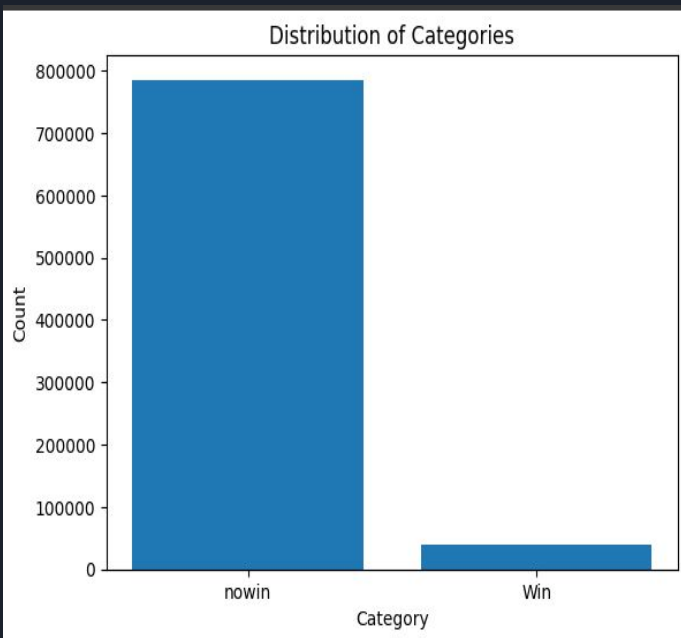

מסקנות מ-value count

סטטיסטיקת ניצחונות/הפסדים: השחקן הפסיד יותר ידיים (152,757) מאשר ניצח (138,044), מה שמסקף את היתרון של הקזינו. היו גם 30,037 תוצאות תיקו (Push).

שכיחות בלאק ג'ק: רק כ-4.78% מהידיים היו בלאק ג'ק טבעי (21 בשני הקלפים הראשונים).

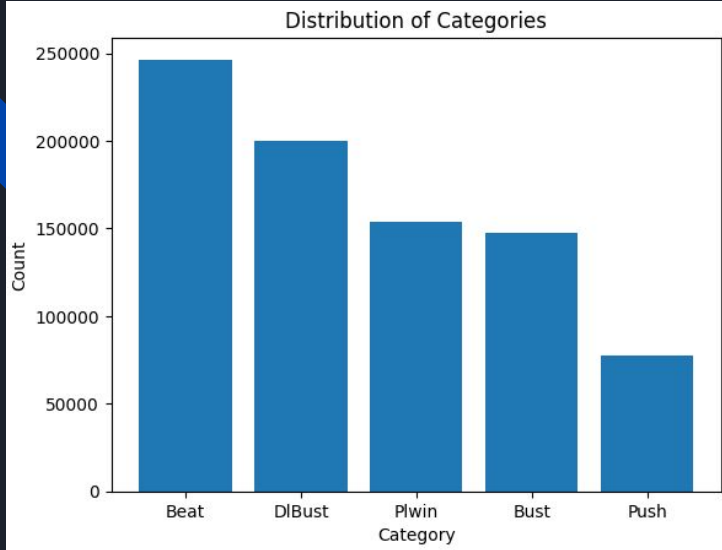
הסכום הנפוץ ביותר הוא **20**, מה שמצביע על כך שרבים מקבלים 10+10, קלפים עם ערך 10 נפוצים בגלל שיש 10 נסיך מלכה ומלך ששווים 10.

גרפים: משתנה יחיד



הגרף הראשון היה על אם השחקן קיבל בלאק'ק וניצח, המסכנה מגרף זה היא שהסיכויים לנצח, גם אם יש לך בלאק'ק הם לא מובטחים ובכללי הסיכויים לבלאק'ק מאוד נמוכים.

מכן אנו מבינים שלא ממולץ לנסות לכוון ל21אם יש לך מספר גבוהה כבר, כי הסיכויים להצליח גם לקבל 21 וגם לנצח הם ממש נמוכים



גרפים: משתנה יחיד

הגרף שפה מראה את המצבים שבהם הסתיים המשחק.

beat- לדילר היה מספר יותר גבוה

dlbust- הדילר עבר את ה21

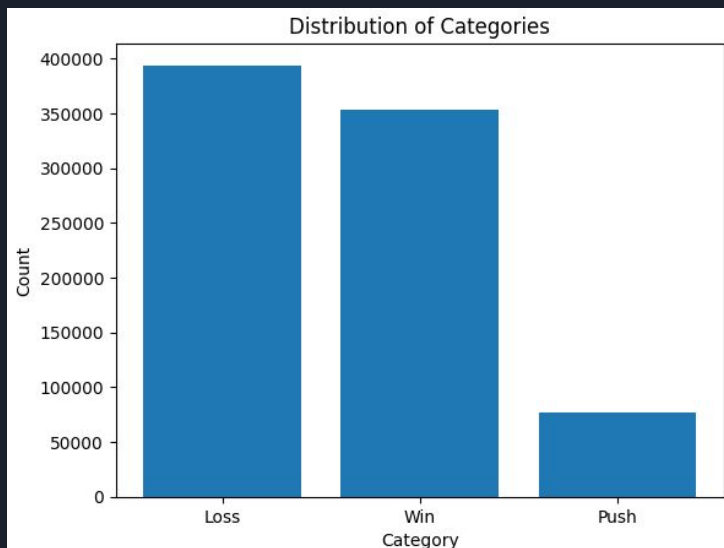
plwin- השחקן קיבל מעל הדילר

bust- השחקן עבר את ה21

push- תיקו

מכן אפשר לראות כי רוב הפעמים לדילר יש מספר יותר גבוהה מלשחקן ולא רק זה גם רוב הפעמים שהשחקן מנצח זה בגלל שהדילר עבר את ה21. עכשיו השחקן מנצח יותר פעמים ממה שהוא עובר את ה21, הדבר הזה אומר לנו שיש סיכוי שהשחקן לא האחד שבאמת משנה התוצאת קלפים שלו. בסופו של דבר רוב הזמן הדילר מקבל יותר ממנו, אמנם אם תהיה לשחקן תוצאה יותר גבוהה הסיכויים ירדו, אך הסיכון גבוהה, ואם רוב הניצחונות של השחקן מגיעות מהדילר עובר את ה21, השחקן לא אמור לדחוף למספרים יותר גבוהים (ברגע שהגיע לכמות מסויימת כמובן).

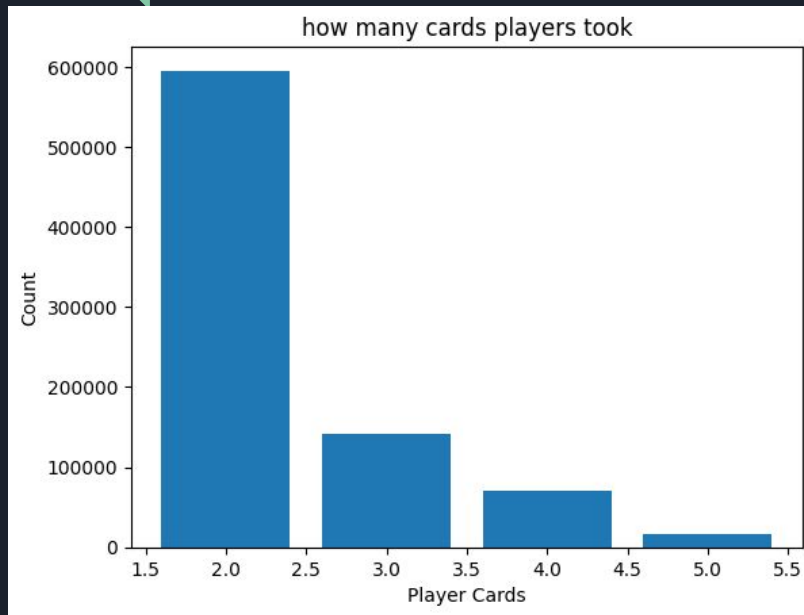
גרפים: משתנה יחיד



הגרף הזה מראה את מספר הנצחונות, הפסדים ותיקוים שקרו בכל היידים.

לפי איך שזה נראה המשחק נוטה לכיוון הדילר אך לא בהרבה מאוד. אז מפה המסקנה היא שהסיבה שהבית מרוויח כל כך במשחק הזה היא בגלל התיקו, הרי אם יש תיקו אז אף אחד לא מנצח והשחקן לא מרוויח כסף. עכשיו התיקו יכול להביא שחקן שיש לו יד של 20 (שזה המספר בעל סיכויים גבוהים) למצב שהוא לא מרוויח כסף והוא בעצם לא בייתרון רב ותמיד שם את השחקן במצב שבו הוא בסופו של דבר יפסיד כסף.

גרפים: משתנה בדיד

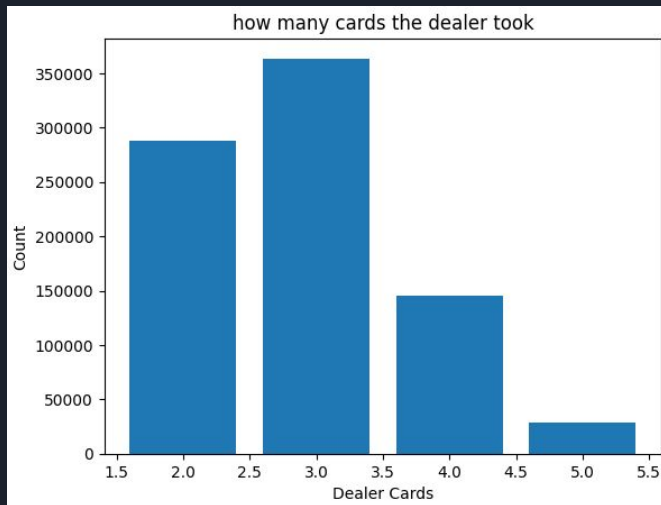


הגרף הזה מראה כמה קלפים השחקנים לקחו.

מכאן אפשר לראות שרוב השחקנים נשארו עם 2 הקלפים הראשונים שלהם. מה שאומר שרובם או

קיבלו תוצאה גבוהה, או מפחדים לקחת עוד קלף בחשש לעבור את ה-21

גרפים: משתנה בדיד



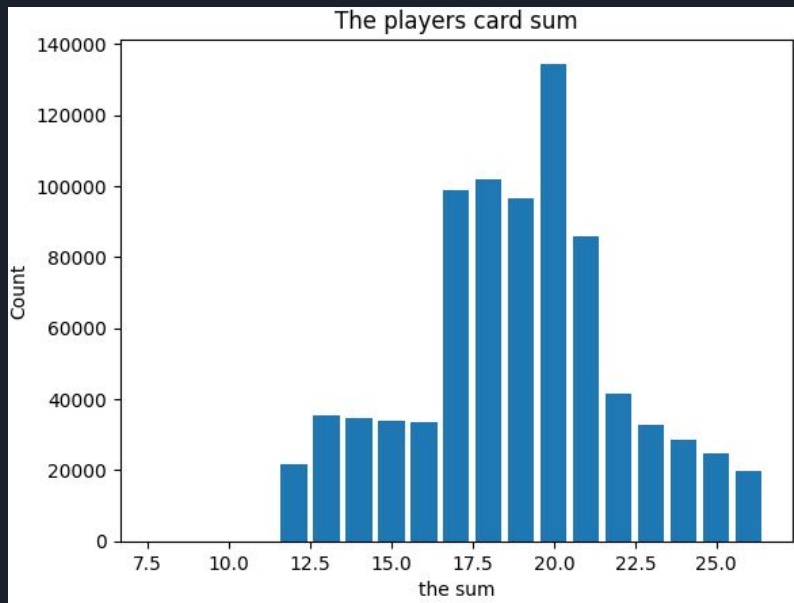
בגרף הזה רואים כמה ידיים הדילר לקח. אפשר לראות לפי הגרף כי הדילר לרוב לוקח

עוד קלף בנוסף ל2 הראשונים. דבר זה מקנה לו את הסיכוי האמיתי לעקוף את היד של

השחקן. עכשיו לפי הגרפים הראשונים אפשר לראות כי הדילר מנצח לרוב לפי יד גבוהה,

מכאן מגיעה השאלה למה שהשחקן אז לא יקח כמעט תמיד עוד קלף? והסיבה היא שהדילר צריך לעקוף או להשוות, ואם הוא כבר יודע את התוצאה של השחקן, ואין לו מה להפסיד אם יש לו מתחת ליד של השחקן.

גרפים: משתנה בדיד

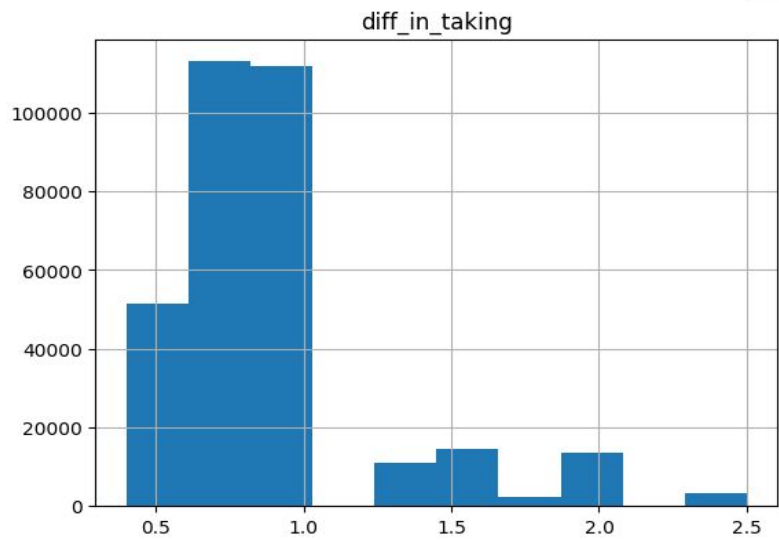
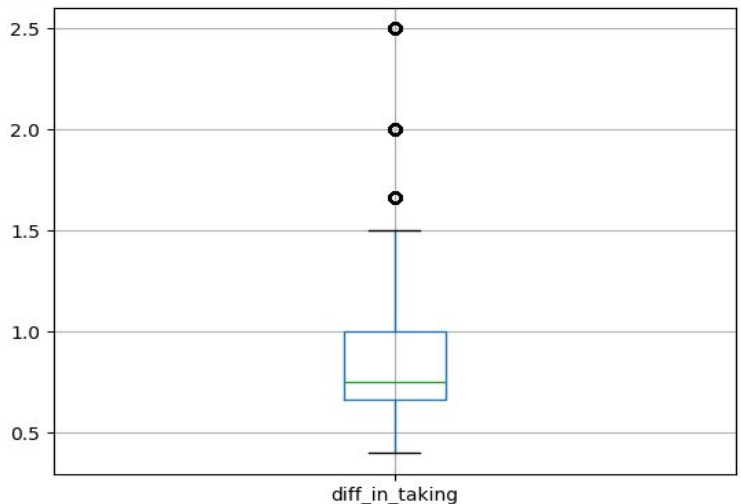


אפשר לראות שרוב הידיים מסתיימות ב20 ובמספרים גבוהים. וגם שהסיכויים ל21 מאוד נמוכים ויש יותר שעברו את ה21 מאשר קיבלו 21 מה שאומר שלא כדאי להיות חמדן ולהסתפק במספר גבוהה ולא הכי גבוהה.

גרפים: משתנה רציף

אפשר לראות שברוב הפעמים השחקן לוקח את אותה כמות הקלפים כמו הדילר

אם לא פחות, מה שמראה שהשחקן יותר מפחד לקחת קלפים מהדילר.

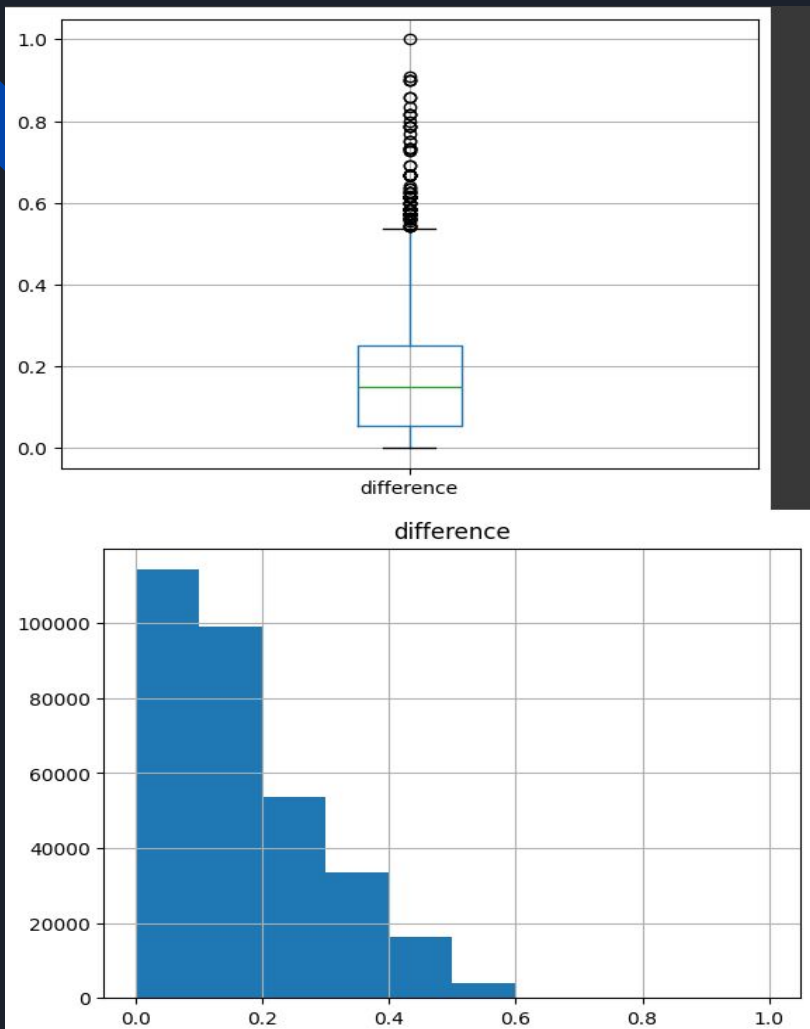


גרפים: משתנה רציף

בגרף הזה $\text{difference} = 0$ אומר שהמספרים שווים
ומשם זה בכמה אחוזים הם שונים

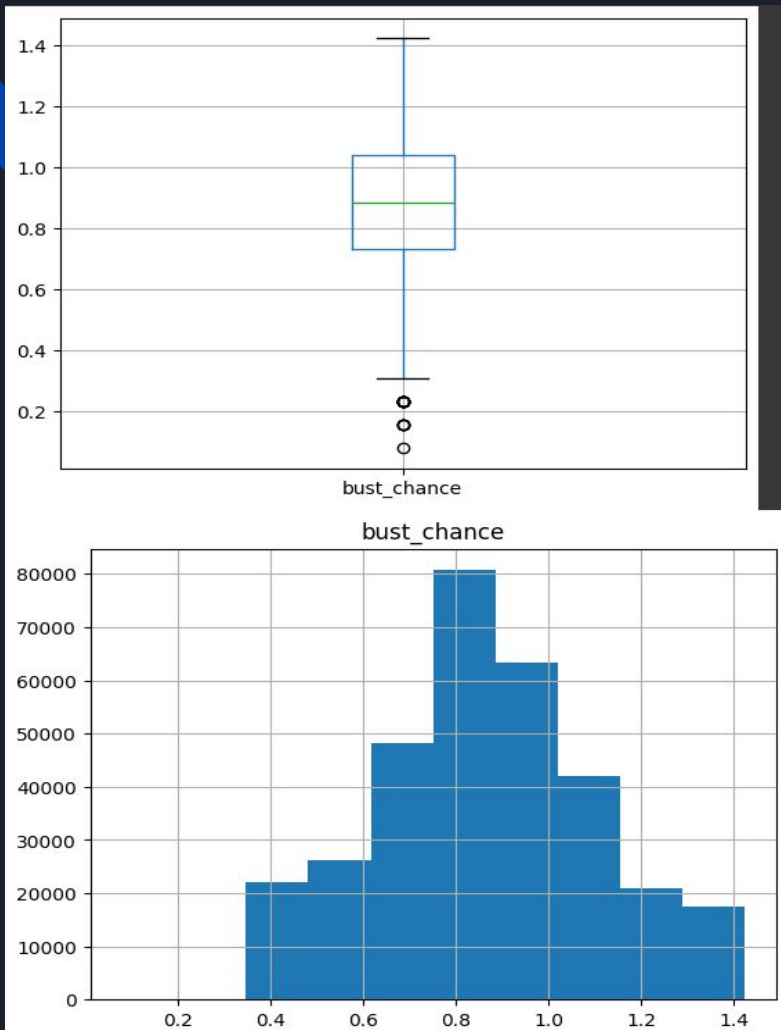
לפי הגרף אפשר לראות שהכי שכיח זה תיקו, מה
שתומך בטענה שהמשחק רווחי בעיקר

בגלל התיקו, ושרוב הזמן התוצאות מאוד קרובות
אחת לשניה.

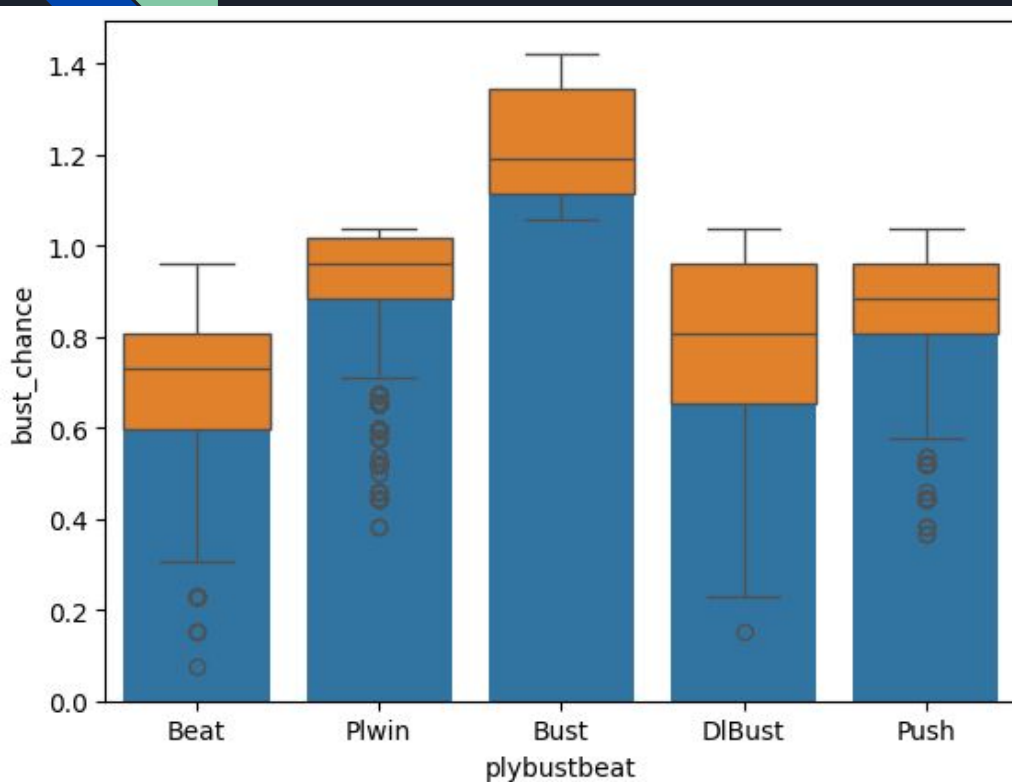


גרפים: משתנה רציף

בגרף הזה אפשר לראות שרוב האנשים
מפסיקים לקחת את הכלפים כהסיכוי
להפסיד הוא 0.8 ומעלה. אך עדיין יש
אנשים שמפחדים ולא לוקחים את הסיכון
גם כשהם בייתרון.



גרפים: משתנה קטגורי ורציף

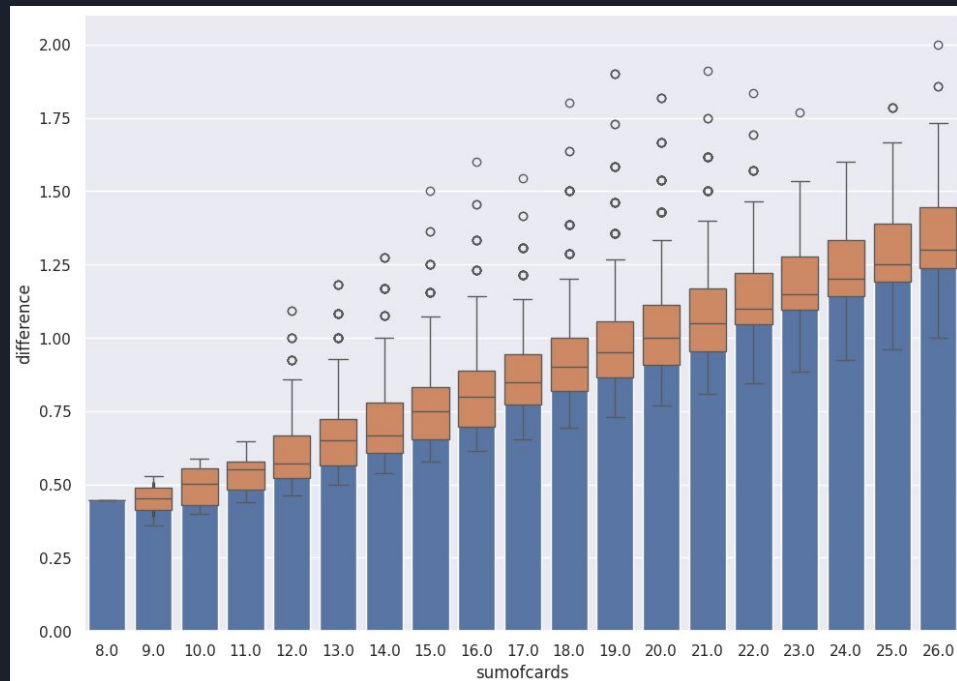


בגרף אפשר לראות כי רוב השחקנים שנצחו את הדילר היו עם מספרים מאוד גבוהים ואלו שעשו את התיקו גם כן.

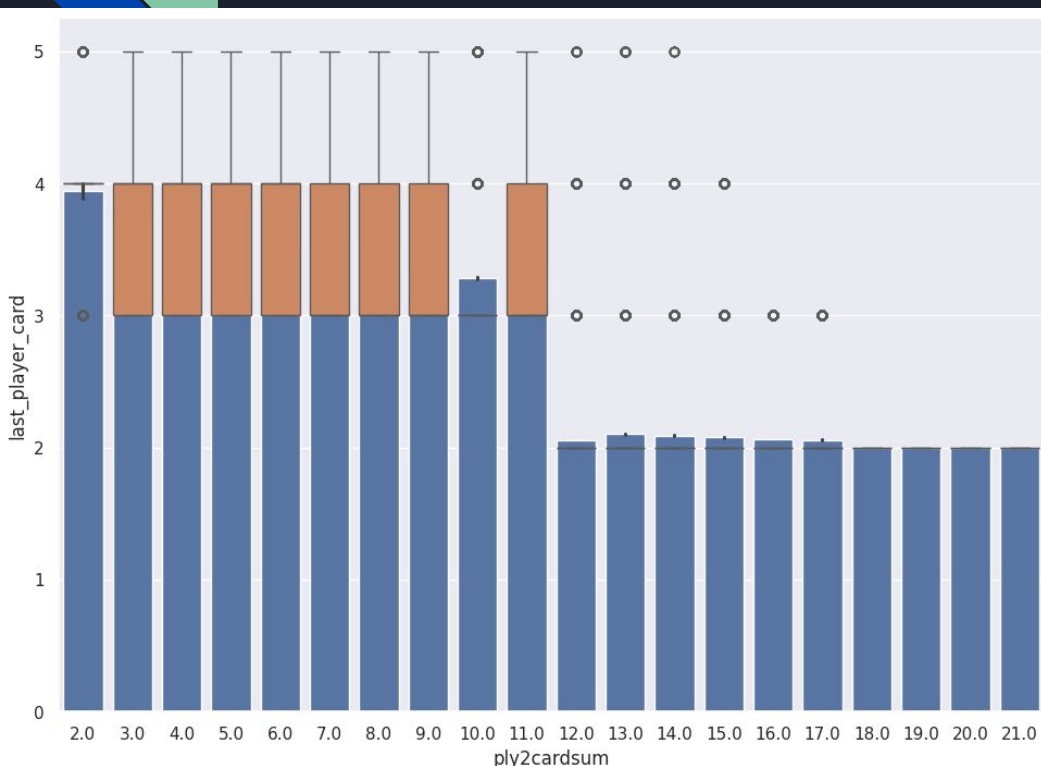
גם אפשר לראות שבבמוצע הדילר עושה באסט לא בהכרח כשהמספרים הכי גבוהים ושם רוב הנצחונות בכללי, ששוב, תומך באמירה שלא לדחוף למספר הכי גבוהה.

גרפים: משתנה בדיד ורציף

הגרף פה מראה כי ככל שהשחקן קיבל מספרים יותר גבוהים כך ההבדל בין הקלפים שלו לשל הדילר נהיים יותר ויותר קרובים, לרמה שלרוב כשהם מקבלים 20 הם רק מגיעים לתיקו בממוצע, שאומר שלא חייב לדחוף למספר הכי גבוהה אם יש לך מספר טוב כבר, כי גם האיזור הממוצע נשאר די אותו הדבר לאורך כל המספרים הגבוהים.



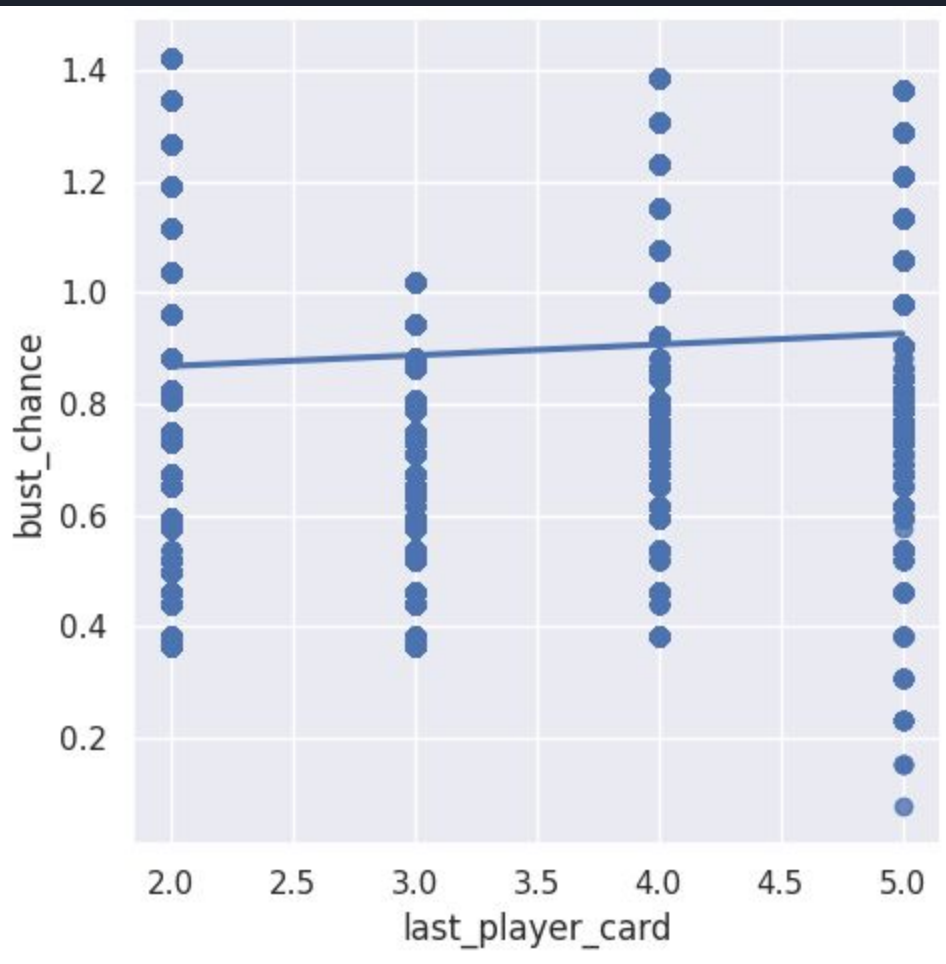
גרפים: משתנה בדיד ורציף



בגרף אפשר לראות את מספר הקלפים הממוצע לפי היד
ההתחלתית של השחקן.

אפשר לראות שבצורה מוזרה מי שמקבל 12 לא לוקח קלף
למרות שהסיכויים לתובתו. גם אפשר לראות שמו שמקבל 9
בממוצע לוקח 4 קלפים ולא מתפשר על מספר נמוך

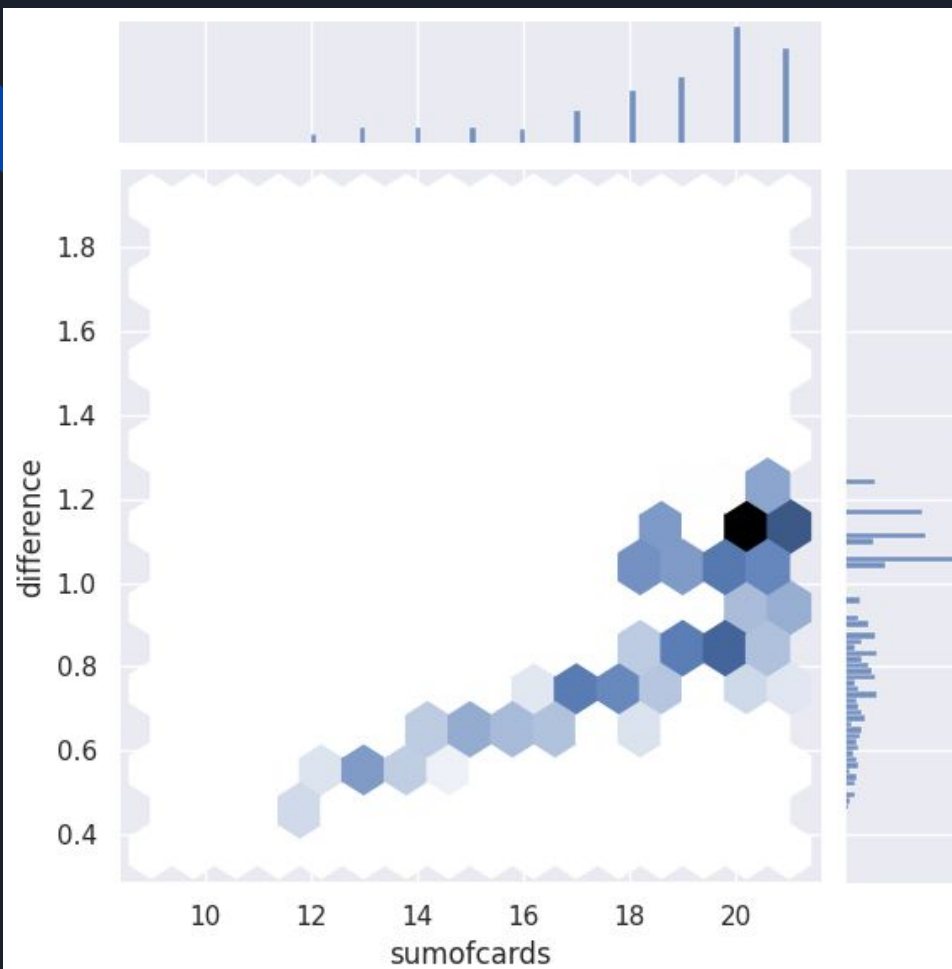
גרפים: 2 רציפים



בגרף הזה רואים את כמות הקלפים ששחקן לקח לסיכויים שלו להפסיד אם הוא לוקח עוד קלף.

ומה שמפתיע פה זה שהסיכויים לא באמת משתנים... הרי היית מצפה.

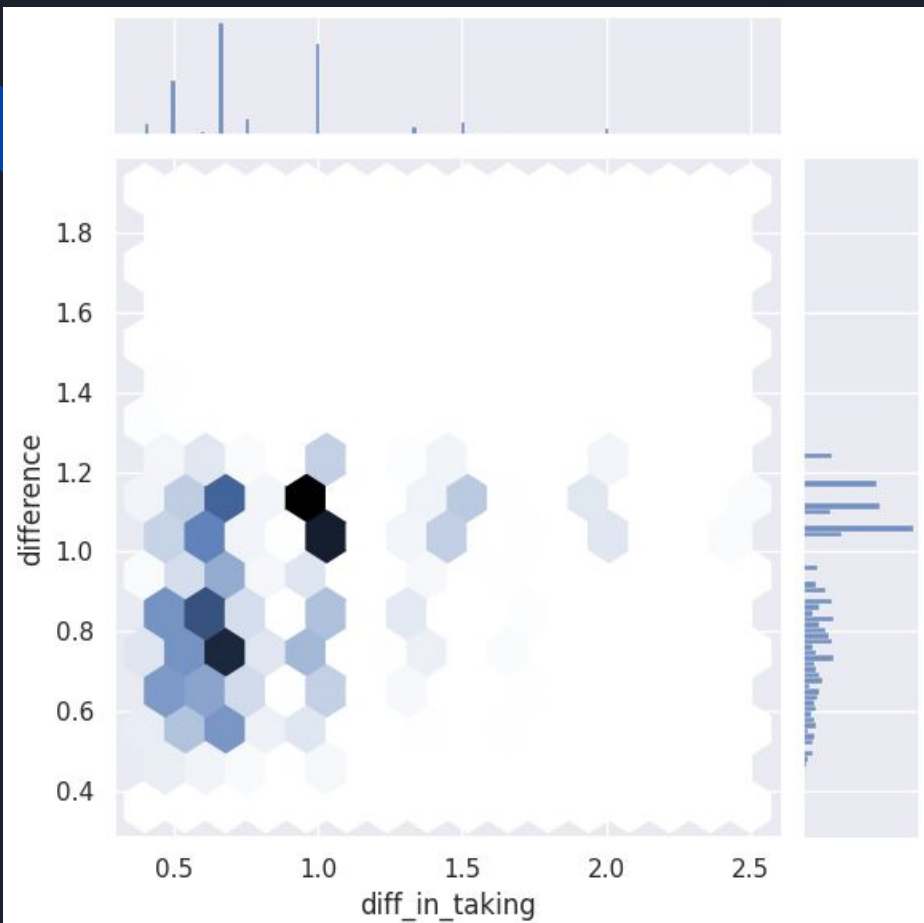
גרפים: 2 משתנים רציפים



בגרף הזה אפשר לראות מה הבדל בין מספר הקלפים לפי מספר הקלפים של השחקן.

אפשר לראות כי מ17 עד 21 רוב הסיכויים לקבל מעל הם אותו הדבר עם 21 בעל סיכוי רב יותר. אך הסיכויים לקבל 21 מאוד נמוכים מה שאומר שאם יש לשחקן מעל 17 לא לקחת עוד קלף כי הסיכון לא שווה את זה

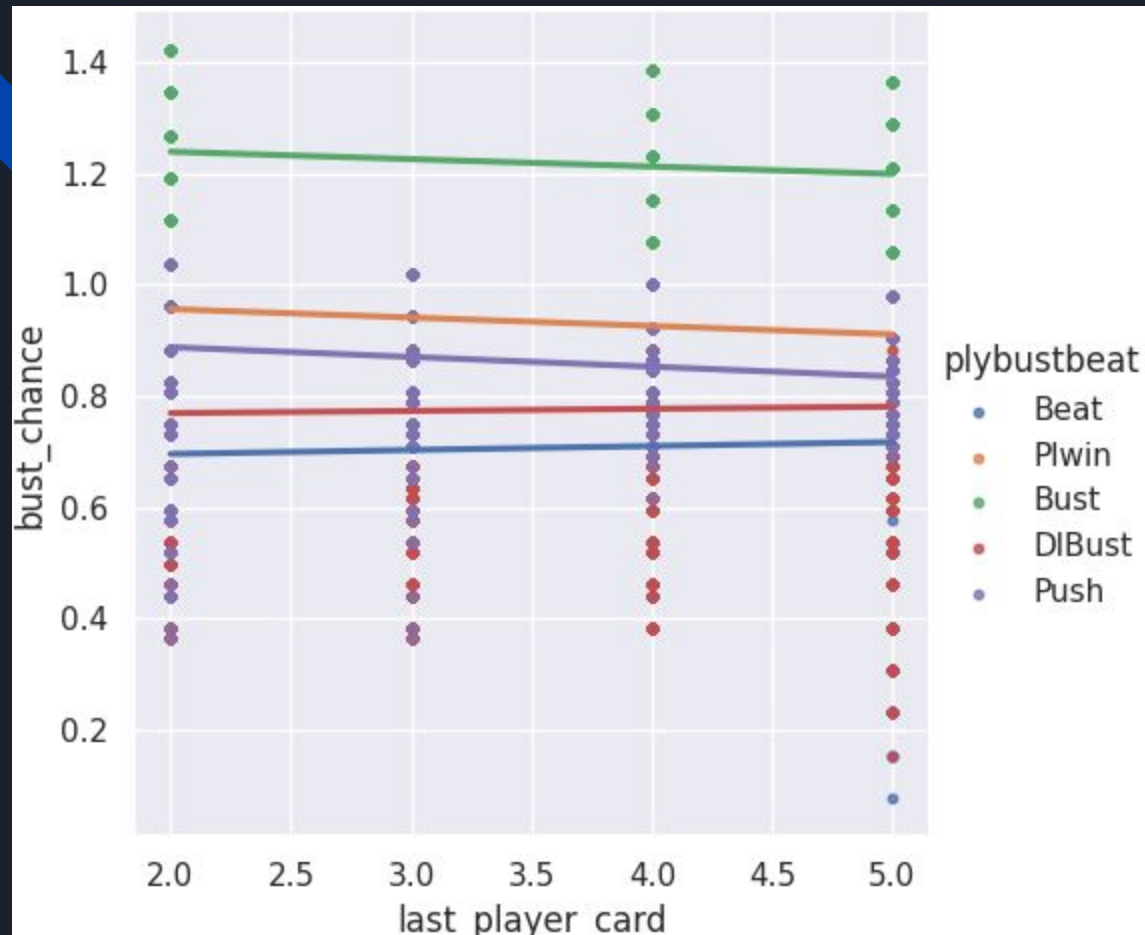
גרפים: 2 משתנים רציפים



בגרף פה אפשר לראות מה ההבדל בין הקלפים לרוב לפי ההבדל בין כמה קלפים השחקן לקח לדילר.

מפה אפשר לראות שרוב הזמן אם השחקן והדילר לקחו אותם כמות קלפים לשחקן יש יתרון מסיים לשחקן וגם אם השחקן לקח פחות קלפים מהדילר זה עדיין עומד מה שמראה לנו שגם מבחינת לקיחת הקלפים, לא כדאי להיות מאוד חמדן.

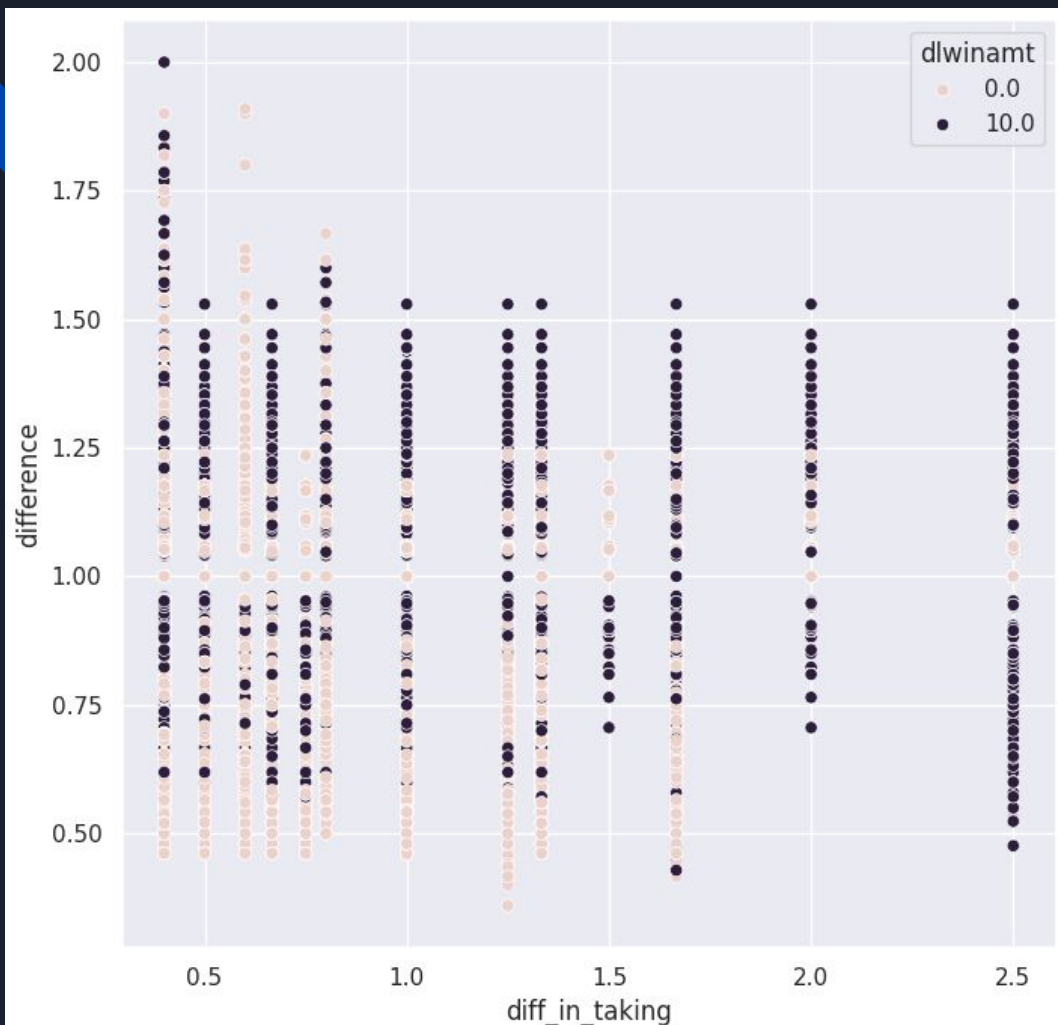
גרפים



הגרף הזה מראה משהוא מעניין הרי היינו מצפים שכל שלוקחים יותר קלפים הסיכויים לניצחון / הפסד משתנים אך בצורה ששונה מהצפייה אין שינוי אמיתי.

בהתחלה זה יכול לגרום לפקפק בקטע של לא להיות חמדן אך אם חושבים על הסיבה למה לקחו קלפים מבינים שלקחו קלפים רק כי היה מספר נמוך וזאת הסיבה למה ה bust chance לא משתנה

גרפים



כאן אפשר גם לראות גם הדילר ניצח או לא ולפי הגרף אפשר חרדות כי הסיכויים הכי גדול לנצח זה על ידי לקוחה של פחות או כמו הדילר ולא להיות חמדן לפי מה שרואים



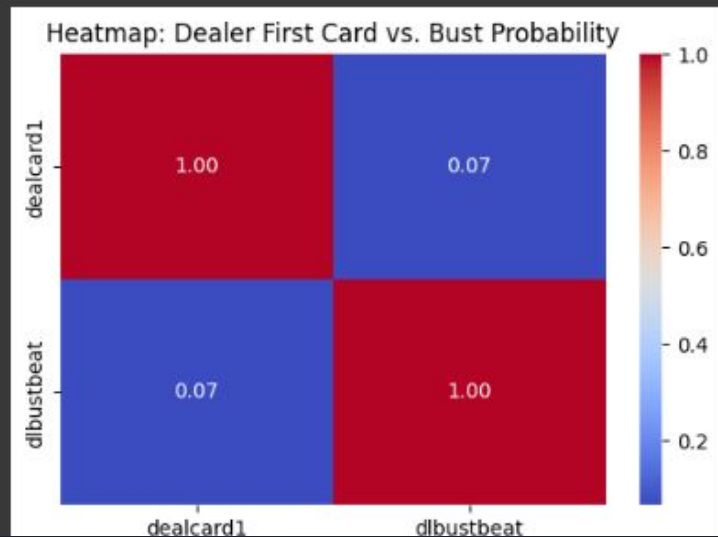
רגרסיה לינארית.

המשתנה התלוי שבחרנו הוא ***dlbustbeat*** (שכפי שציינו מסמן אם הדילר חרג מ21). הוא יהיה התוצאה שנרצה לנבא באמצעות המשנה הבלתי תלוי שלנו.

המשתנה הבלתי תלוי שלנו הוא ***dealcard1*** (שהוא הקלף הראשון של הדילר). בחרנו בו בשביל לנבא את הסיכוי שהדילר יחרוג מ21 בגלל שהוא הגורם הראשוני שקובע איך ימשיך המשחק וניתן לראות אותו בתחילת המשחק.

הצגת מסקנות ממפת החום

```
# Keep only relevant columns
df = df[["dealcard1", "dlbustbeat"]]
# Convert `dlbustbeat` to numeric
df["dlbustbeat"] = df["dlbustbeat"].apply(lambda x: 1 if x == "Beat" else 0)
# Compute correlation
correlation_matrix = df.corr()
# Create the heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Heatmap: Dealer First Card vs. Bust Probability")
plt.show()
```



ביצענו ניתוח נתונים כדי לבדוק האם הקלף הראשון של הדילר משפיע על הסיכוי שלו לחרוג.

מפת החום שהפקנו מציגה את מקדמי המתאם בין שני המשתנים: המתאם שהתקבל הוא 0.07 בלבד, כלומר אין קשר חזק בין הקלף הראשון של הדילר לבין הסיכוי שלו לחרוג.

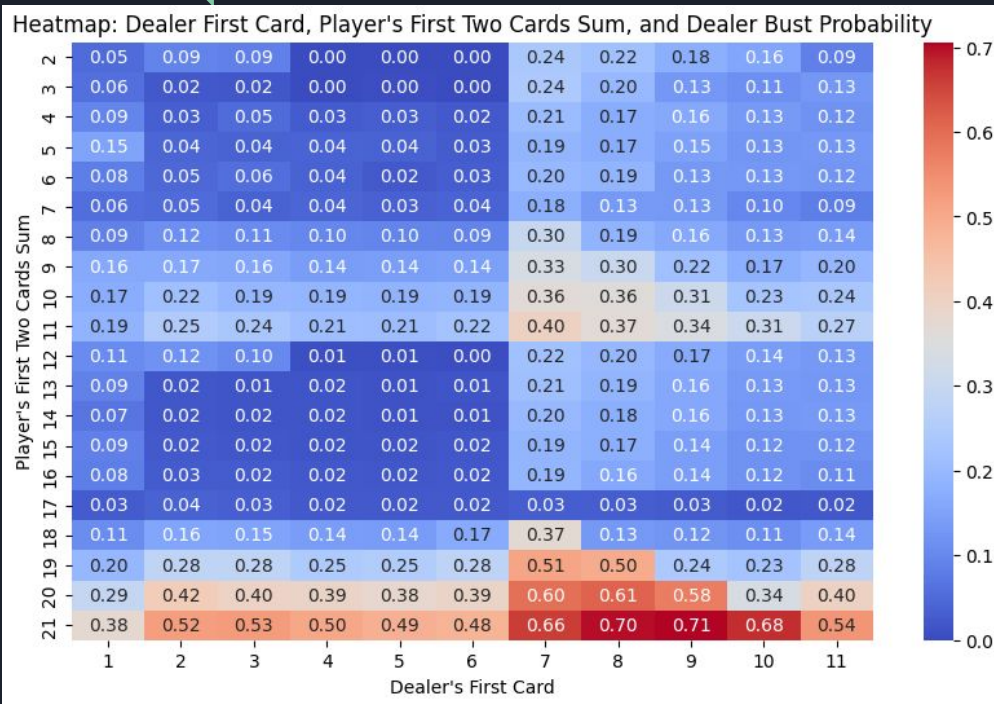
המשמעות היא שהקלף הראשון לא מספק מידע מועיל מספיק כדי לחזות אם הדילר יפסיד.

הסבר אפשרי לממצאים

המהלכים של השחקן משפיעים על תוצאת המשחק: למשל, אם השחקן בוחר לעצור מול קלף נמוך של הדילר, הדבר עשוי להשפיע על המשך המשחק ועל סיכויי הדילר לחרוג.

חידוד המסקנה באמצעות משתנים נוספים

ניתוח יותר מעמיק בעמוד הבא.



בגלל שהשתמשנו במשתנה אחד כדי להגיע למסקנה, נחدد אותה עם משתנה נוסף (סכום 2 הקלפים הראשונים של השחקן).

כיצד זה מחדד את המסקנה הראשונית?
בתחילה חשבנו שהקלף הראשון של הדילר לא משפיע משמעותית על תוצאות המשחק. עכשיו אנחנו רואים שהוא כן משפיע, אך רק בשילוב עם הנתונים של השחקן. המשמעות היא שהשחקן לא יכול להסתמך רק על קלף הפתיחה של הדילר, אלא צריך לשקלל גם את מצבו האישי במשחק.



סיכום הממצאים ממפת החום

בתחילת הבדיקה מצאנו שהקשר בין קלף ראשון של הדילר לבין הסיכוי שלו לחרוג הוא נמוך מאוד (0.07). המשמעות היא שלבדו, הקלף הראשון של הדילר לא מספק מידע מספיק כדי לחזות אם הדילר יפסיד. בהוספת *ply2cardsum* (סכום שני הקלפים הראשונים של השחקן), זיהינו דפוסים מעניינים: כאשר לשחקן יש 20 או 21 נקודות, הסיכוי של הדילר לחרוג גבוה יותר. כאשר לשחקן יש סכום נמוך מ-10, הסיכוי של הדילר לחרוג נמוך מאוד, ללא קשר לקלף הראשון שלו. הסיכוי של הדילר לחרוג גבוה יותר שהקלף הראשון הוא 6, 7 או 8. כאשר הדילר פותח עם קלפים אלו, אחוז ההפסדים שלו מגיע ל-60-70%, במיוחד כאשר לשחקן יש סכום גבוה.

זהו חיזוק לטקטיקה ידועה בבלאקג'ק, לפיה כדאי לתקוף יותר כאשר הדילר פותח עם 6 או 7. הקלפים 2-5 של הדילר נחשבים "ניטרליים" מבחינת סיכון לחריגה. למרות שנהוג לחשוב שדילר עם 2-5 הוא בעייתי, בפועל הסיכוי שלו לחרוג נמוך מ-50% ברוב המצבים. המשמעות היא שאין צורך למהר לתקוף מול קלפים אלו.

ממצא מפתיע – קלף ראשון של 9 אצל הדילר מוביל לפחות הפסדים (של הדילר) מהצפוי. בניגוד להנחה הרווחת, כאשר הדילר מתחיל עם 9, אחוזי ההפסד שלו נמוכים יחסית.

חלוקה לנתוני אימון ונתוני מבחן

בשלב זה חילקנו את הנתונים לשתי קבוצות:

נתוני אימון (60%) – משמשים ללימוד.

נתוני מבחן (40%) – משמשים להערכת הדיוק.

החלוקה מתבצעת באופן רנדומלי תוך שמירה על עקביות (random_state=101).

הפלט מציג את גודל מערכי האימון והמבחן: 540,000 שורות לאימון ו-360,000 שורות למבחן.

```
# Select relevant columns for regression
df = df[["dealcard1", "ply2cardsum", "dlbustbeat"]]
X = df[["dealcard1", "ply2cardsum"]]
y = df["dlbustbeat"]

# Split data into training (60%) and testing (40%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=101)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(540000, 2)
(360000, 2)
(540000,)
(360000,)
```

אימון המודל באמצעות נתוני האימון

משוואת הישר שקיבלנו:

$$y = -0.1197 + 0.0088D + 0.0171P$$

כאשר:

Y - ההסתברות שהדילר יחרוג.

D - הקלף הראשון של הדילר.

P - סכום שני הקלפים הראשונים של השחקן.

```
from sklearn.linear_model import LinearRegression

lm = LinearRegression()
# Train the model on the training set
lm.fit(X_train, y_train)
# Display regression coefficients
print("Regression Coefficients:", lm.coef_)
# Display intercept
print("Intercept:", lm.intercept_)

# Create a DF to show coefficient
cdf = pd.DataFrame(lm.coef_, X_train.columns, columns=['Coeff'])
print(cdf)
```

```
Regression Coefficients: [0.00882277 0.01709578]
Intercept: -0.1196852870359407
```

	Coeff
dealcard1	0.008823
ply2cardsum	0.017096

חישוב R^2 ו RMSE

$$R^2 = 0.04$$

המודל ממש גרוע בלהסביר את הנתונים, כלומר
המשתנים שנבחרו אינם מנבאים היטב את הסיכוי של
הדילר לחרוג.

$$RMSE = 0.38$$

שגיאה הממוצעת גבוהה יחסית, מה שמראה שהחזוי
אינו מדויק.

```
from sklearn.metrics import mean_squared_error
import numpy as np

# Predict values for the test set
predictions = lm.predict(X_test)

# Calculate R^2
r2 = lm.score(X_test, y_test)
print("R^2:", r2)

# Calculate RMSE
rmse = np.sqrt(mean_squared_error(y_test, predictions))
print("RMSE:", rmse)
```

```
R^2: 0.040441601143124006
RMSE: 0.38157185659629805
```



מסקנות מהרגרסיה הליניארית

- המודל נבנה כדי לחזות האם הדילר יחרוג לפי הקלף הראשון של הדילר וסכום שני הקלפים של השחקן.
- מקדמי המתאם מראים שלסכום הקלפים של השחקן יש השפעה גדולה יותר על חיזוי ההפסד של הדילר מאשר לקלף הראשון של הדילר.
- מדדי הדיוק ($R^2 = 0.0044$, $RMSE = 0.38$) מצביעים על כך שהמודל אינו מסביר את הנתונים היטב.
- **מסקנה:** הקלף הראשון של הדילר וסכום הקלפים של השחקן אינם מספיקים כדי לנבא האם הדילר יחרוג.

רעיונות לשיפור:

- בדיקת משתנים נוספים כמו מספר הקלפים שהדילר משך או סכום היד שלו.
- ניסיון להשתמש ברגרסיה לוגיסטית שמתאימה יותר לחיזוי משתנים בינאריים (חרג או לא חרג).



הצגת תשובות לשאלת המחקר

שאלת המחקר: האם ניתן לחזות את סיכויי ההפסד של הדילר בבלאקג'ק באמצעות קלף הפתיחה שלו וסכום שני הקלפים הראשונים של השחקן?

תשובות:


קלף הפתיחה של הדילר לבדו אינו מספיק כדי לחזות בצורה מדויקת את הסיכוי שלו להפסיד. התוספת של סכום שני הקלפים הראשונים של השחקן משפרת את הדיוק בחיזוי, במיוחד כאשר הסכום גבוה (20 או 21).



מסקנות משניות ממאגר הנתונים

נמצא מתאם חיובי בין קלף הפתיחה של הדילר לבין סיכוי ההפסד כאשר הקלפים הם 6, 7 או 8: הסיכוי של הדילר לחרוג גבוה יותר כשהקלף הראשון הוא אחד מאלו.

השפעת המהלכים של השחקן: כאשר לשחקן יש סכום נמוך מ-10, הסיכוי של הדילר לחרוג נמוך מאוד, ללא קשר לקלף הראשון שלו.



התאמת הממצאים לאסטרטגיות קיימות במשחק בלאקג'ק

שיטת ספירת קלפים

מה אומרת השיטה?

שיטה זו מתמקדת ביחס בין קלפים גבוהים לנמוכים שנותרו בחפיסה, ככל שיש יותר קלפים גבוהים (10, אס), הסיכוי שהדילר יחרוג גבוה יותר.


מה גילינו?

מצאנו כי קלף הפתיחה של הדילר אינו משפיע משמעותית על הסיכוי לחרוג. זה מחזק את הרעיון שהרכב החפיסה כולה חשוב יותר מאשר קלף בודד.

מסקנה

הממצאים תומכים בגישה של ספירת קלפים, שמדגישה את החשיבות של ניתוח הרכב החפיסה ולא קלף בודד.

מקור והסבר על ספירת קלפים - https://www.youtube.com/watch?v=G_So72IFNIU



הימורי צד על חריגת הדילר (Side bets)

מה אומרת האסטרטגיה?

ישנן גרסאות של בלאק'ק המאפשרות להמר על חריגת הדילר (לדוגמה, Side Bet על קלף שלישי).

מה גילינו?

מאחר והקלף הראשון של הדילר אינו מנבא טוב חריגה, הימורי צד כאלה מבוססים על מזל ולא על ניתוח מושכל. הסתברות לזכות בהימור כזה נמוכה.

מסקנה:

הממצאים אינם תומכים בהימורי צד על בסיס קלף ראשון בלבד, ומרמזים שהימור כזה פחות משתלם.

מקור והסבר על הימורי צד - <https://news.williamhill.com/casino-guides/blackjack-side-bets>