Unfair Game

טל חסון ויצחק יעקובוב

הקדמה:

בתור חובבי משחקי מחשב, החלטנו לבדוק האם ניתן לחזות משחקים מצליחים בהתבסס על קריטריונים שונים. כמו כן, רצינו לחקור האם ישנן חברות או פלטפורמות בעלות יתרון משמעותי בתחום.

השתמשנו במקור הבא: Metacritic reviews

Crawling

חילקנו את עבודתנו בשלב זה לשני חלקים:

- בשלב הראשון עברנו על כל האתר, משכנו עבור כל משחק את ה-href הרלוונטי עבורו וריכזנו אותם בקובץ Csv.

Scraping all href links in 1 page

```
def get_hrefs_list(url):

    result = requests.get(url, headers=HEADERS)
    soup = BeautifulSoup(result.content, "html.parser")

    list_wrapper = soup.find_all("div", class_="browse_list_wrapper")

    links_list = []
    for wrapper in list_wrapper:
        for link in wrapper.find_all("a", href = True, class_="title"):
            links_list.append(link['href'])

    return links_list
```

This function is for getting an href links from a given pages range

Every url for each page is in the same structure: 'BASE URL?#num of page'

```
def get_hrefs_list_fromPage_toPage(url, begin = 0, end = 0):
    href_list = []
    for page in range(begin, end + 1):
        if(page != 0):
            print("page: ",page)
            new_url = f"{url}?page={page}"
        else:
            new_url = url
        href_list.extend(get_hrefs_list(new_url))
    return href_list
```

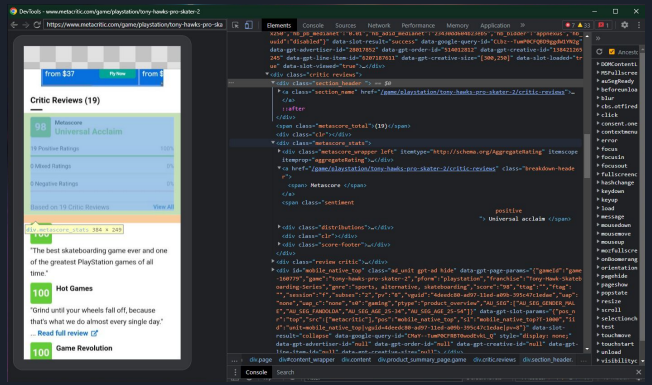
Crawling

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19223 entries, 0 to 19222
Data columns (total 16 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   game_name               19223 non-null  object
 1   platform                19223 non-null  object
 2   publisher               19211 non-null  object
 3   release_date            19223 non-null  object
 4   meta_scroe              19214 non-null  float64
 5   user_score              19220 non-null  object
 6   develeoper              19202 non-null  object
 7   genres                  19223 non-null  object
 8   num_of_players          15494 non-null  object
 9   rating                  17111 non-null  object
 10  user_positive_review    17950 non-null  object
 11  user_negative_review    17950 non-null  object
 12  user_mixed_review       17950 non-null  object
 13  critic_positive_review  19216 non-null  float64
 14  critic_negative_review  19216 non-null  float64
 15  critic_mixed_review     19216 non-null  float64
dtypes: float64(4), object(12)
memory usage: 2.3+ MB
```

חילקנו את עבודתנו בשלב זה לשני חלקים:

- בשלב השני קראנו מתוך קובץ ה-Csv שיצרנו בשלב הקודם והתחלנו להרכיש את הנתונים והמאפיינים השונים של כל משחק.

בחלק זה התבצעה גם הרכשת נתונים בעזרת המובייל



- 'HEADERS_MOBILE' - cell phone user agent
- 'HEADERS' - computer user agent

```
# user-agents with every request
HEADERS = {'User-Agent': 'Mozilla/5.0 (iPad; CPU OS 12_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML, li
HEADERS_MOBILE = { 'User-Agent' : 'Mozilla/5.0 (iPhone; CPU iPhone OS 9_1 like Mac OS X) AppleWebKit/60
```

Data handling

- טיפול בנתונים חסרים וחריגים

Dropping the nan values for: 'meta_scroe', 'publisher', 'develeoper':

This information can not be determined and replaced

```
df.dropna(subset=['meta_scroe'], inplace=True)
df.dropna(subset=['publisher'], inplace=True)
df.dropna(subset=['develeoper'], inplace=True)
```

We will assign to each rating a value related to the order of the rating

```
df["rating"].unique()
```

```
array(['E', 'T', 'M', 'E10+', nan, 'K-A', 'AO', 'RP'], dtype=object)
```

First we replace the nan valuse with the top value:

```
top_val = df["rating"].describe()[2]
df["rating"].fillna(top_val, inplace=True)
```

The values will be distributed like this:

- RP - Rating Pending == 0
- E - Everyone 6 and older == 1
- E10+ - 10 and older == 2
- K-A - 10 and older == 2
- T - Teen 13 and older == 3
- m - 17+ == 4
- AO - Adults Only 18 years and older == 5

```
rating_replace_map = {
    'RP': 0, 'E': 1, 'E10+': 2,
    'K-A': 2, 'T': 3, 'M': 4, 'AO': 5}

df["rating"].replace(rating_replace_map, inplace=True)
```

changing the 'tbd' to nan values:

```
df.replace("tbd", np.nan, inplace=True)
df[df["user_score"] == "tbd"]
```

```
df.dropna(subset=['user_score'], inplace=True)
```

Conversion of all review data to int type without the commas

```
colums = ["user_positive_review", "user_negative_review", "user_mixed_review"]
for column in colums:
    nums = []
    for string in df[column]:
        if(type(string) != float):
            nums.append(int(string.replace(",", "")))
        else:
            nums.append(np.nan)
    df[column] = nums
```

Data handling

- מחיקת כפילויות ואיחוד בין הנתונים

Game name duplications:

Handling duplicates in 'game_name'

```
duplicates = df[df["game_name"].duplicated(keep=False)].copy()
duplicates_names = list(duplicates["game_name"].unique())
df.reset_index(drop=True, inplace=True)

for duplicate_name in duplicates_names:
    duplicates_game = df[df["game_name"] == duplicate_name]

    # saving the mean valu of the duplicates
    meta_score_mean = duplicates_game["meta_score"].mean()
    user_score_mean = duplicates_game["user_score"].mean()
    critic_mean = duplicates_game["critic_positive_normelize"].mean()
    user_mean = duplicates_game["user_positive_normelize"].mean()

    row_index = df[df["game_name"] == duplicate_name].index[0]
    df.loc[row_index, "meta_score"] = meta_score_mean
    df.loc[row_index, "user_score"] = user_score_mean
    df.loc[row_index, "critic_positive_normelize"] = critic_mean
    df.loc[row_index, "user_positive_normelize"] = user_mean
  # saving the earliest date of the duplicates
  year_month_zip = zip(duplicates_game["release_year"], duplicates_game["release_month"])
  release_year = min(duplicates_game["release_year"])
  release_month = 12
  for year, month in year_month_zip:
      if(year == release_year and month < release_month):
          release_month = month
  df.loc[row_index, "release_year"] = release_year
  df.loc[row_index, "release_month"] = release_month

df = df.drop_duplicates(subset="game_name")
df.reset_index(drop=True, inplace=True)
```

לפני

After Duplicates Handling

```
df[df["game_name"] == "Assassin's Creed Origins"]
```

	game_name	platform	genres	develeoper	publisher
1223	Assassin's Creed Origins	PC, PlayStation	action adventure, edutainment, miscellaneous, ...	Ubisoft	Ubisoft

אחרי

```
df[df["game_name"] == "Assassin's Creed Origins"]
```

	game_name	platform	genres	develeoper	publisher
1674	Assassin's Creed Origins	PC	action adventure, edutainment, miscellaneous, ...	Ubisoft	Ubisoft
3227	Assassin's Creed Origins	PlayStation	action adventure, edutainment, miscellaneous, ...	Ubisoft	Ubisoft

Genres duplications:

לפני

```
df["genres"][23]
```

```
'Action, Arcade, First-Person, Sci-Fi, Sci-Fi, Shooter, Shooter'
```

Handling Duplicates:

```
genres_per_game = []
for game_genres in df["genres"]:
    genre_list = set(game_genres.strip(", ").lower().split(", "))
    genre_list = sorted(list(genre_list))
    genres_new_string = ", ".join(genre_list)
    genres_per_game.append(genres_new_string)

df["genres"] = genres_per_game
```

```
len(df["genres"].unique())
```

```
1371
```

אחרי

```
df["genres"][23]
```

```
'action, arcade, first-person, sci-fi, shooter'
```

Data handling

- חילוץ נתונים ויצירת עמודות חדשות

making 'exclusive_game' column:

Content:

1 - Games released for only one platform

0 - Otherwise

```
exclusive_games_list = []
for platforms in df["platform"]:
    platforms_list = platforms.split(", ")
    if(len(platforms_list) != 1 or 'Old Platform' in platforms_list):
        exclusive_games_list.append(0)
    else:
        exclusive_games_list.append(1)
df['exclusive_game'] = exclusive_games_list
```

The 'release_date' will be divided into two columns: 'release_month' and 'release_year'

```
df["release_date"].unique()
```

```
array(['Nov 23, 1998', 'Sep 20, 2000', 'Apr 29, 2008', ...,
       'Apr 25, 2001', 'Jul  6, 2006', 'Dec 21, 2011'], dtype=object)
```

```
import calendar
```

```
# Getting months by their numbers
month_number = {}
for index, month in enumerate(calendar.month_abbr):
    if(month):
        month_number[month.lower()] = index

# Divides into two columns:
release_date_list = list(df["release_date"])
release_year = []
release_month = []
for date in release_date_list:
    year = re.findall(r'\b\d+\b', date)[1]
    release_year.append(int(year))
    month = re.findall(r'(\w\w\w) ', date)[0]
    release_month.append(month_number[month.lower()])

df['release_year'] = release_year
df['release_month'] = release_month
```

```
df.drop("release_date", axis=1, inplace=True)
```

Data handling

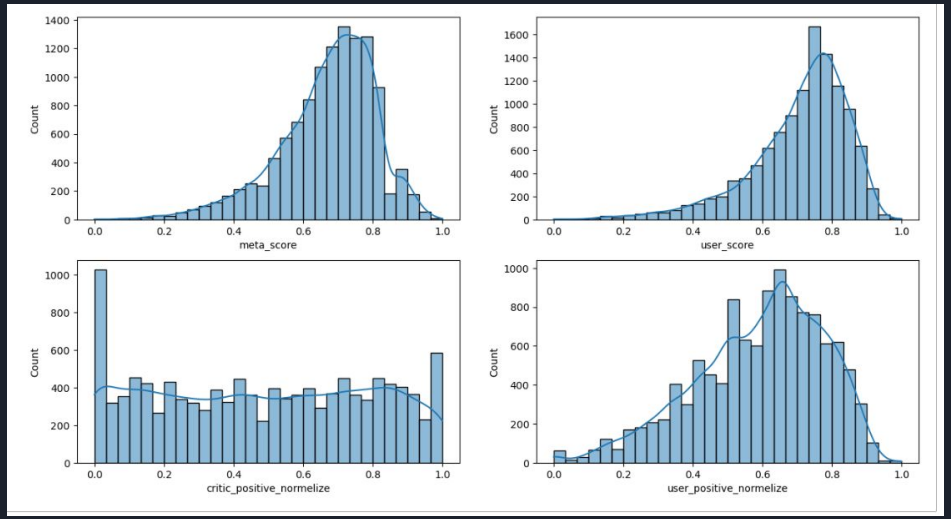
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11695 entries, 0 to 11694
Data columns (total 19 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   game_name                   11695 non-null  object
 1   platform                    11695 non-null  object
 2   genres                      11695 non-null  object
 3   develeoper                  11695 non-null  object
 4   publisher                   11695 non-null  object
 5   max_players                 11695 non-null  int64
 6   online_game                 11695 non-null  int64
 7   release_year                11695 non-null  int64
 8   release_month               11695 non-null  int64
 9   rating                      11695 non-null  int64
 10  meta_score                  11695 non-null  float64
 11  user_score                  11695 non-null  float64
 12  critic_positive_normelize   11695 non-null  float64
 13  user_positive_normelize     11695 non-null  float64
 14  exclusive_game              11695 non-null  int64
 15  publisher_labeled           11695 non-null  int32
 16  develeoper_labeled          11695 non-null  int32
 17  genres_labeled              11695 non-null  int32
 18  platform_labeled            11695 non-null  int32
dtypes: float64(4), int32(4), int64(6), object(5)
memory usage: 1.5+ MB
```

- לאחר סיום הטיפול בנתונים נכון לשלב זה:

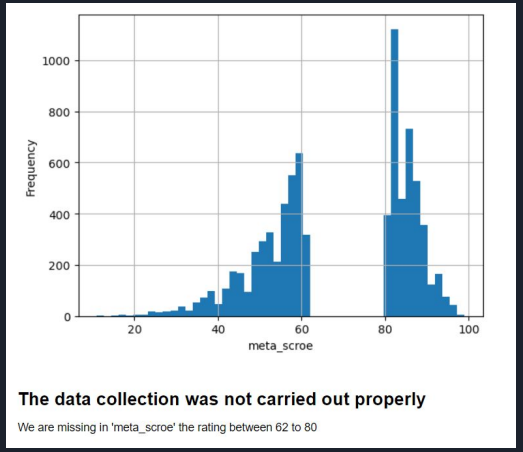
game_name	platform	genres	develeoper	publisher	max_players	online_game	release_year	release_month	rating	meta_score	user_score	critic_positive_normelize	user_positive_normelize	exclusive_game	publisher_labeled	develeoper_labeled	genres_labeled	platform_labeled
The Legend of Zelda: Ocarina of Time	Nintendo	action adventure, fantasy	Nintendo	Nintendo	1	0	1998	11	1	1.000000	0.936170	1.000000	0.893962	1	1070	2078	262	0
Tony Hawk's Pro Skater 2	Nintendo, Old Platform, PC, PlayStation	alternative, skateboarding, sports	Neversoft Entertainment	Activision	2	0	2000	9	3	0.931818	0.765957	0.970909	0.717737	0	59	2045	805	3
Grand Theft Auto IV	PC, PlayStation, Xbox	action adventure, modern, open-world	Rockstar North	Rockstar Games	1	0	2008	4	4	0.958333	0.780142	0.983333	0.681603	0	1323	2572	323	43
SoulCalibur	Old Platform, Xbox	3d, action, fighting	Namco	Namco	2	0	1999	9	3	0.880682	0.829787	0.947368	0.725062	0	1042	2013	143	38
Super Mario Galaxy	Wii	3d, action, platformer	Nintendo	Nintendo	0	0	2007	11	1	0.977273	0.936170	1.000000	0.911193	1	1070	2078	154	51

EDA

- נתקלנו פה בבעיה שנבעה מאיסוף לא תקין של הנתונים



לאחר הרכשת כל הנתונים מהאתר

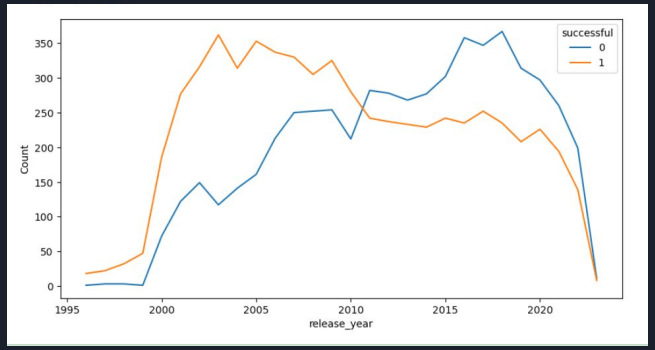


- פתרון: חזרה לשלב ה-Crawling

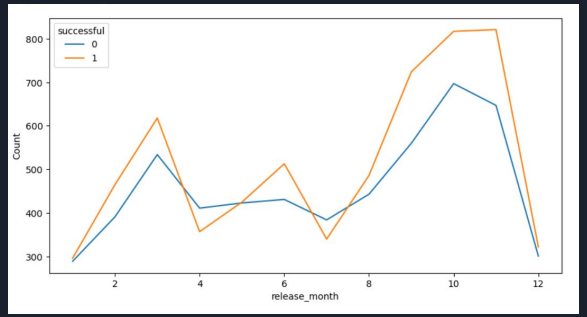
EDA

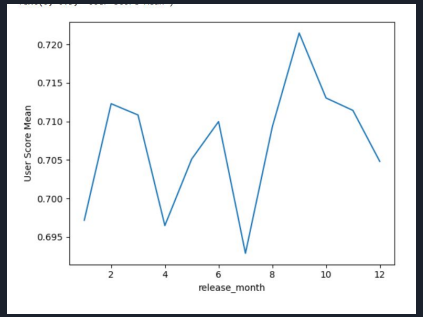
- בדיקת השנים והחודשים בהם המשחקים המצליחים יצאו

בדיקת שנים: בין השנים 1995 ל-2010 יצאו יותר משחקים מצליחים מלא מצליחים



בדיקת חודשים: התקופה הטובה ביותר לשחרר משחק היא בין החודשים ספטמבר לדצמבר, כשהשיא - באוקטובר.



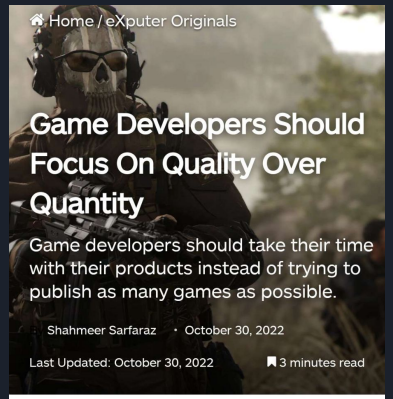


מהיכן נובעת התנהגות זו?

EDA

ציטוט מהרשת:

"Video games quantity over quality"



Today, many video game companies publish the same game with a slight reskin every year. It's like they've made it a goal for themselves to publish a game, even if it offers nothing new except for a few changes in the story. It is about time game developers realized that they should focus on quality over quantity.

Why is gaming in the 90s and Early 2000s viewed as the Golden Era of Gaming?

Not everyone feels this way, and there are still so many great games being made to this day. But even if you have this mindset, you've probably noticed on countless occasions other gamers drooling with nostalgia over games of the olden days. I'd like to have an open discussion as to why this is, and why some of us view gaming as a shadow of what it used to be. I would like to offer my thoughts as to why this is, and maybe we can start from there...

I think what made Video Games so fantastic back in the day was "Innovation through Necessity." Developers didn't have the technology to do most of what they wanted, unlike today. Systems such as the Super Nintendo, the PsONE and the PS2 were limited in what they could do. A lot of developers at the time were capable of creating fun and engaging gameplay with what they had to work with. But in order to make their games special, they had to express themselves through music, atmosphere and story, and they had to do it creatively, so that the player could immerse themselves in a world made of jagged polygons.

ציטוט מהרשת:

"...Most games launch in the fall because they want to cash in on all the holiday shopping…"

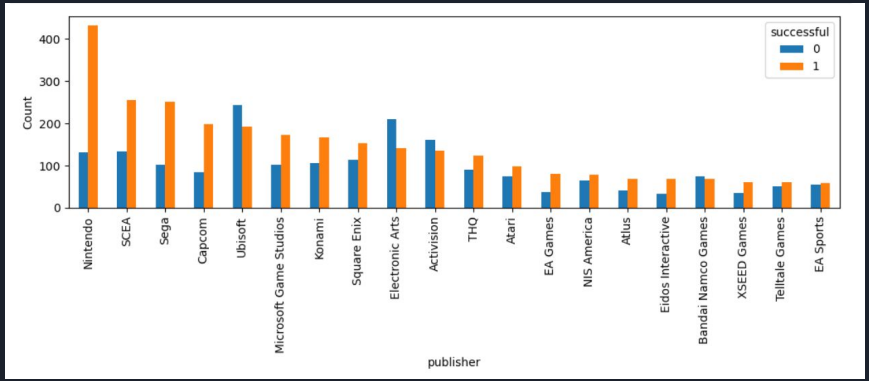
yahoo!

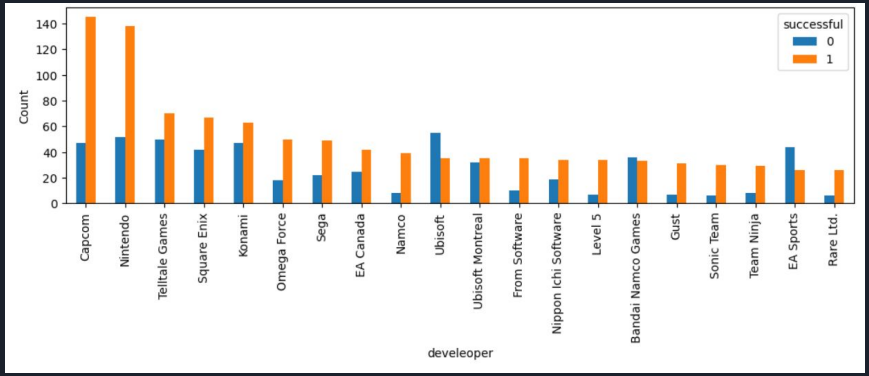
Game companies traditionally release their biggest titles in the lead-up to the holidays to get their hits out during the busiest shopping season of the year.

A man walks an advertisement of Nintendo Switch at an electronics retail chain store in Tokyo on Oct. 13, 2021. (AP Photo/Koji Sasahara)

Last year, the gaming giants fired off a host of big-name titles including "Halo Infinite," "Battlefield 2042," "Metroid Dread," and "Call of Duty Vanguard" to name a few. While "Battlefield" and "Call of Duty" underperformed expectations, the titles were still among the best performing

EDA

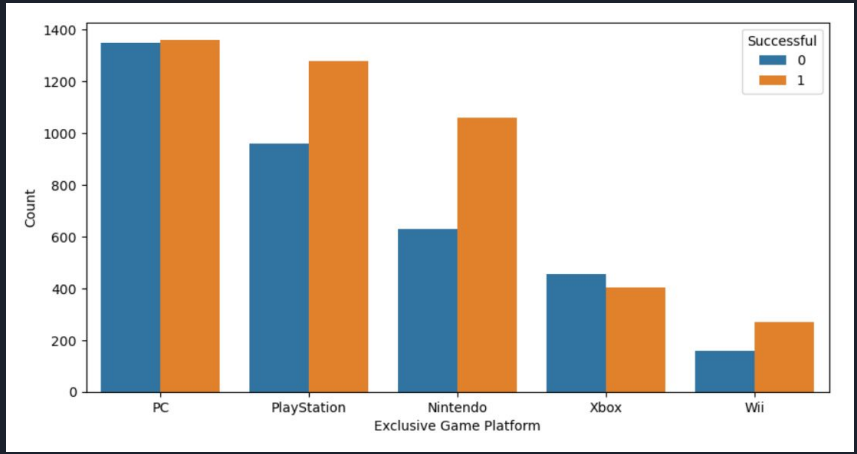




- הצגת כמות המשחקים המוצלחים ביחס לכמות המשחקים הלא מוצלחים בהתבסס על Publisher & Developer - ניתן לראות יתרון מובהק לחברת "Nintendo".

* נציין כי בחרנו להציג את 20 המפתחים והמפיצים המובילים בתעשייה.

EDA



- בחינת המשחקים הייחודיים לפלטפורמות השונות - נשים לב "פלייסטיישן" ו"נינטנדו" מובילות ביחס בין כמות המשחקים המוצלחים לבין אלו שנחשבים ללא מוצלחים (Exclusive only).

Machine Learning

- המודל ללמידת מכונה המרכזי שבחרנו: **Logistic regression**

תוצאות התחלתיות:

Logistic regression:

```
In [466]:
df_for_ml = df.iloc[:, 5:].copy()
df_for_ml["successful"] = (df_for_ml["user_score"] >= 0.73).astype("int64")

y = df_for_ml["successful"]
X = df_for_ml.copy()
to_del = ["user_score", "meta_score", "successful"]
X.drop(to_del, axis=1, inplace=True)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=45)
```

```
In [467]:
log = LogisticRegression(max_iter=500)

log, y_pred = train_and_print(log, X_train, X_test, y_train, y_test, 3, "successful", check_train=True)
```

```
'Successful':

-------------
Test:
accuracy is: 0.526
precision is: 0.53
recall is: 0.4
f1 is: 0.456

confusion matrix:
TP=748 | FN=1122
FP=663 | TN=1236
---------------
---------------
Train:
accuracy is: 0.531
precision is: 0.523
recall is: 0.416
f1 is: 0.463

confusion matrix:
TP=1780 | FN=2503
FP=1621 | TN=2890
-------------
```

קיבלנו תוצאות לא מספקות ולכן חזרנו לשלב הטיפול בנתונים:

1. טיפול שונה בעמודת ה-User score
2. מחיקת כפילויות שלא מצאנו קודם לכן

Machine Learning

- **Logistic regression**

לאחר הטיפול בנתונים:

Accuracy, precision, recall and f1 ⬆

Logistic regression:

```
In [471]: log = LogisticRegression(max_iter=500)

          log, y_pred = train_and_print(log, X_train, X_test, y_train, y_test, 3, "successful", check_train=True)
```

```
'Successful':

-------------
Test:
accuracy is: 0.535
precision is: 0.54
recall is: 0.75
f1 is: 0.628

confusion matrix:
TP=1379 | FN=459
FP=1173 | TN=498
-------------
-------------
Train:
accuracy is: 0.535
precision is: 0.545
recall is: 0.741
f1 is: 0.628

confusion matrix:
TP=3220 | FN=1126
FP=2684 | TN=1156
-------------
```

קיבלנו תוצאות טובות יותר אך רצינו לבחון האם ניתן להמשיך לשפר:

מפני שכבר הרכשנו את כל הנתונים מהאתר וחזרנו לשלב הטיפול בנתונים ושיפורם - החלטנו לנסות מודלים נוספים

Machine Learning

Decision Trees

Accuracy, precision, f1	↑
Recall	↓

Decision Trees

```
In [411]: parameters = {'max_depth':range(2, 100, 2), "min_samples_split":range(2, 50, 2) }
          dt = tree.DecisionTreeClassifier()

          clf = GridSearchCV(dt, parameters,scoring=make_scorer(metrics.accuracy_score, greater_is_better=True))
          clf.fit(X_train, y_train)
          print("best parameter set is:",clf.best_params_," and its score was",clf.best_score_)

          best parameter set is: {'max_depth': 2, 'min_samples_split': 2}  and its score was 0.6086003387774921

In [412]: y_pred_train = clf.predict(X_train)
          y_pred = clf.predict(X_test)
          print("Test DecisionTree:\n")
          print_results( y_test, y_pred, 3)
          print("Train DecisionTree:\n")
          print_results( y_train, y_pred_train, 3)

          Test DecisionTree:

          accuracy is: 0.625
          precision is: 0.632
          recall is: 0.68
          f1 is: 0.655

          confusion matrix:
          TP=1249 | FN=589
          FP=726 | TN=945
          -------------
          Train DecisionTree:

          accuracy is: 0.615
          precision is: 0.627
          recall is: 0.678
          f1 is: 0.651

          confusion matrix:
          TP=2947 | FN=1399
          FP=1756 | TN=2084
          -------------
```

KNN

Accuracy, precision	↑
Recall, f1	↓

k Nearest Neighbors (k-NN):

```
In [399]: to_num = int(math.sqrt(len(y_train)))
          to_num

Out[399]: 90

In [400]: parameters = {'n_neighbors':range(3,to_num,2) }
          knn = KNeighborsClassifier()
          clf = GridSearchCV(knn, parameters, scoring=make_scorer(metrics.f1_score, greater_is_better=True))
          clf.fit(X_train, y_train)

          print("best parameter set is:",clf.best_params_," and its score was",clf.best_score_)

          best parameter set is: {'n_neighbors': 73}  and its score was 0.6146720619480276

In [401]: y_pred = clf.predict(X_test)
          print("Test KNN:")
          print_results( y_test, y_pred, 3)
          y_pred_train = clf.predict(X_train)
          print("Train KNN:")
          print_results( y_train, y_pred_train, 3)

          Test KNN:
          accuracy is: 0.537
          precision is: 0.546
          recall is: 0.685
          f1 is: 0.608

          confusion matrix:
          TP=1259 | FN=579
          FP=1046 | TN=625
          -------------
          Train KNN:
          accuracy is: 0.578
          precision is: 0.585
          recall is: 0.707
          f1 is: 0.64

          confusion matrix:
          TP=3073 | FN=1273
          FP=2181 | TN=1659
          -------------
```

מסקנות:

לאחר ניסיונות רבים, לא הצלחנו לשפר משמעותית את תוצאות החיזוי.
מכאן הסקנו, כי בהסתמך על החומר הנלמד במהלך הקורס, היכולת לחזות האם משחק יצליח בהתבסס על מאפייניו השונים היא מוגבלת.

יש לציין שגילינו דברים מעניינים בשלב ה-EDA:

- בין השנים 1995-2010 כמות המשחקים המצליחים הייתה גדולה יותר בהשוואה לכמות המשחקים הלא מצליחים באותה תקופה, אך בשנים שלאחר מכן המגמה התהפכה.

- שמנו לב כי התקופה הטובה ביותר להוציא משחקים היא תקופת החגים(בחו"ל) - בין החודשים ספטמבר לדצמבר.

- הבחנו בעליונות מסוימת של חברות פיתוח והפצה כמו "Nintendo" ו-"Capcom" ביחס לחברות אחרות. כמו כן, שמנו לב להבדלים בכל הקשור למשחקים ייחודיים(Exclusive) והיתרון שקיים ל-"Playstation" ול-"Nintendo" בתחום.