How AI can grow the Gaming Industry - Tushar arora

Problem Statement

The gaming industry is not big in some countries like India but they grow fastly. There are lots of problems in the gaming industry that can be solved by artificial intelligence. For example:-

- Video Games Sales & trend prediction
- Recommendation for a new game or product.
- Spam Detection in reviews.
- Market and gamer's category segmentation.
- Fighting Hate Speech and Trolls classification(Analysing chat in-game)
- Sensitivity Finder.
- Creating Bots and AI Games.
- Anti-cheat detection.
- Make a cluster in an open-world game.
- To get an overview of Self-driving cars.

1. Video Games Sales & trend prediction

Video game sales analysis is a popular problem statement on Kaggle. We can work on this problem to analyze the sales of more than 16,500 games or we can also train a machine learning model for forecasting video game sales. These game data can be given by video game digital distribution service and using this you can grow the business of video game digital distribution services using Artificial intelligence. Video game digital distribution giants like Steam, rockstar have been using affinity analysis to perform Market Basket Analysis, which identifies purchasing habits of customers and uses this information to cross-sell and up-sell relevant items.

- **Market/Customer/Business needs Assessment:-** Our goal is to predict the revenue that is going to be generated by those potential customers in the near future. The customer buying preferences have been significantly changed due to the pandemic. Therefore, by using this technique, we aim to provide sales forecasting with useful insights from the available data and ways to generate more revenue.

- **Target Specification:-** The proposed system/service will provide the video game digital distribution with some techniques so that their sales boost up and they no longer have to go through an economic crisis.

- **External Search:-** Kaggle Competition
  https://www.kaggle.com/gregorut/videogamesales?select=vgsales.csv

- **Applicable Patents:-** Data analysis tools of python and machine learning algorithms to predict sales in the future.

- **Applicable Regulations:-**
    - Data protection and privacy regulations(Customers)
    - Employment Laws
    - Regulations against false advertising.

- **Applicable Constraintsble:-** Continuous data collection and maintenance, Computer power according to data and focus on rarely bought products.

- **Business Opportunity:-** Since the above technique has only been used by large companies because of data but we can grow our own small business using all over market data analysis.

- **Concept Generation:-** Come from the recommendation system with some analysis.

- **Concept Development:-** Basically we need to predict sales in the future and Using analysis we can make a strategy to get more sales.

- **Final Product Prototype (abstract) with Schematic Diagram:-**
    - Collect data from Kaggle.
    - Analyzing.
    - Preprocessing.
    - Training.
    - Model creation & deployment.

- We use pandas, seaborn, NumPy, matplotlib, and Plotly for analysis of the data.
- Data contain 16598 rows x 11 columns
- Columns:-
    - Ranking -- Game ranking based on the total sales (in millions)
    - Name -- Name of the Game
    - Platform -- Game Platforms like (PS4, PC, GB, etc)
    - Year -- Year of game release
    - Genre -- Simply the game genre (sports, racing ...)
    - publisher -- the name of the publisher
    - NA_Sales -- Sales in North America (in millions)
    - EU_sales -- Sales in Europe (in millions)
    - JAP_sales -- Sales in Japan (in millions)
    - Global_Sales -- Total sales worldwide (in millions)

- Spreading of data:-

	Rank	Year	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
count	16598.000000	16327.000000	16598.000000	16598.000000	16598.000000	16598.000000	16598.000000
mean	8300.605254	2006.406443	0.264667	0.146652	0.077782	0.048063	0.537441
std	4791.853933	5.828981	0.816683	0.505351	0.309291	0.188588	1.555028
min	1.000000	1980.000000	0.000000	0.000000	0.000000	0.000000	0.010000
25%	4151.250000	2003.000000	0.000000	0.000000	0.000000	0.000000	0.060000
50%	8300.500000	2007.000000	0.080000	0.020000	0.000000	0.010000	0.170000
75%	12449.750000	2010.000000	0.240000	0.110000	0.040000	0.040000	0.470000
max	16600.000000	2020.000000	41.490000	29.020000	10.220000	10.570000	82.740000

- Check unique category in columns:-

```
In [10]:
x = videogame_df['Name'].unique() #using numpy.ndarray to
y = videogame_df['Genre'].unique()
z = videogame_df['Publisher'].unique()
z = videogame_df['Publisher'].unique()
```

```
In [11]:
print('Total Games by `Name` count(unique) :',len(x))
print('Total Games by `Genre` count(unique) :',len(y))
print('Total Games by `Publisher` count(unique) :',len(z))
```

```
Total Games by `Name` count(unique) : 11493
Total Games by `Genre` count(unique) : 12
Total Games by `Publisher` count(unique) : 579
```

- Number of platform and genre.

```
In [13]:
df['Platform'].value_counts().to_dict().keys()
```

```
dict_keys(['DS', 'PS2', 'PS3', 'Wii', 'X360', 'PSP', 'PS', 'PC', 'XB', 'GBA', 'GC', '3DS', 'PSV', 'PS4
', 'N64', 'SNES', 'XOne', 'SAT', 'WiiU', '2600', 'NES', 'GB', 'DC', 'GEN', 'NG', 'SCD', 'WS', '3DO', 'T
G16', 'GG', 'PCFX'])
```

```
In [17]:
df['Genre'].value_counts().to_dict().keys()
```

```
dict_keys(['Action', 'Sports', 'Misc', 'Role-Playing', 'Shooter', 'Adventure', 'Racing', 'Platform', 'S
imulation', 'Fighting', 'Strategy', 'Puzzle'])
```

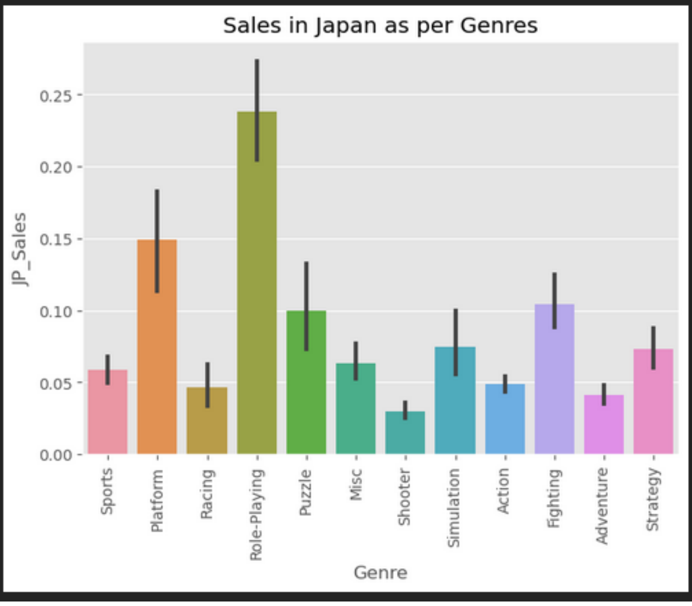
- Check null values and handle missing values.

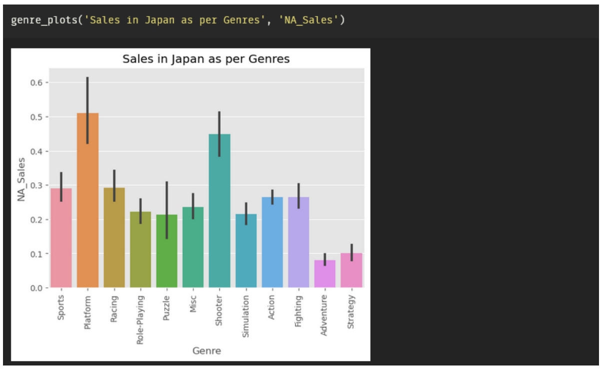
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16291 entries, 0 to 16597
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Rank          16291 non-null  int64  
 1   Name          16291 non-null  object 
 2   Platform      16291 non-null  object 
 3   Year          16291 non-null  float64
 4   Genre         16291 non-null  object 
 5   Publisher     16291 non-null  object 
 6   NA_Sales      16291 non-null  float64
 7   EU_Sales      16291 non-null  float64
 8   JP_Sales      16291 non-null  float64
 9   Other_Sales   16291 non-null  float64
 10  Global_Sales  16291 non-null  float64
dtypes: float64(6), int64(1), object(4)
memory usage: 1.5+ MB
```

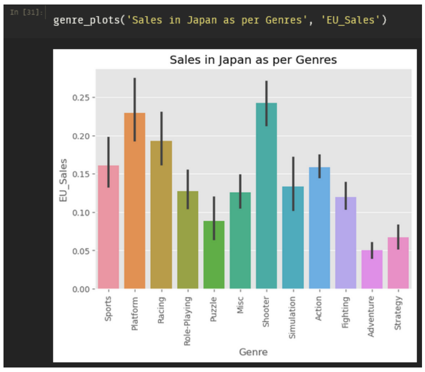
```
In [4]: len(df[df['Year'].isnull() | df['Publisher'].isnull()])/ len(df) * 100

1.8496204361971322

In [5]: # Missing values contain 1 persent that way i am removing.
        df.dropna(inplace=True)
```

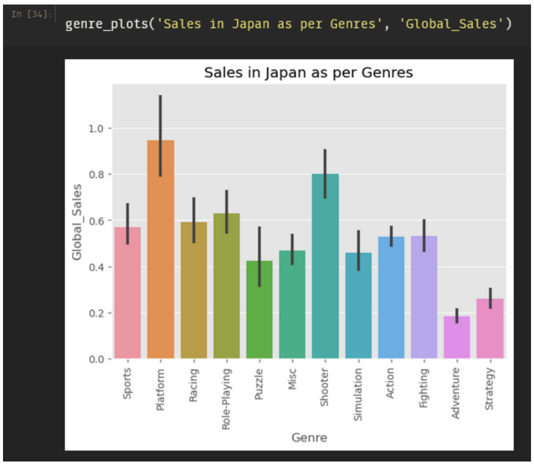
- Sales in Different region according to the genre.

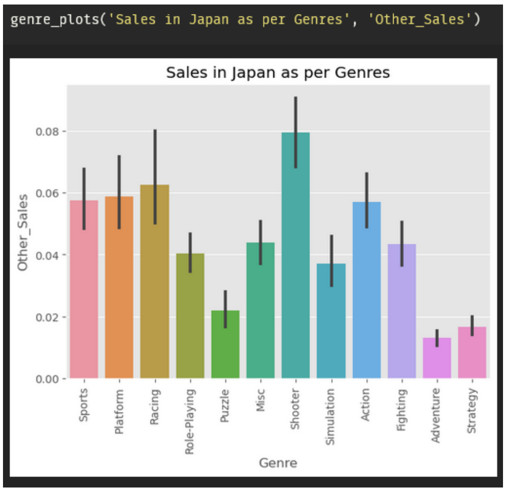
```
In [30]: # function to plot graphs
         def genre_plots(title, sales):
           plt.figure(figsize = (7,5), dpi= 100)
           plt.title(title)
           plt.xticks(rotation = 90)
           sns.barplot(x = 'Genre',
                       y = sales,
                       data = df, )
         genre_plots('Sales in Japan as per Genres', 'JP_Sales')
```





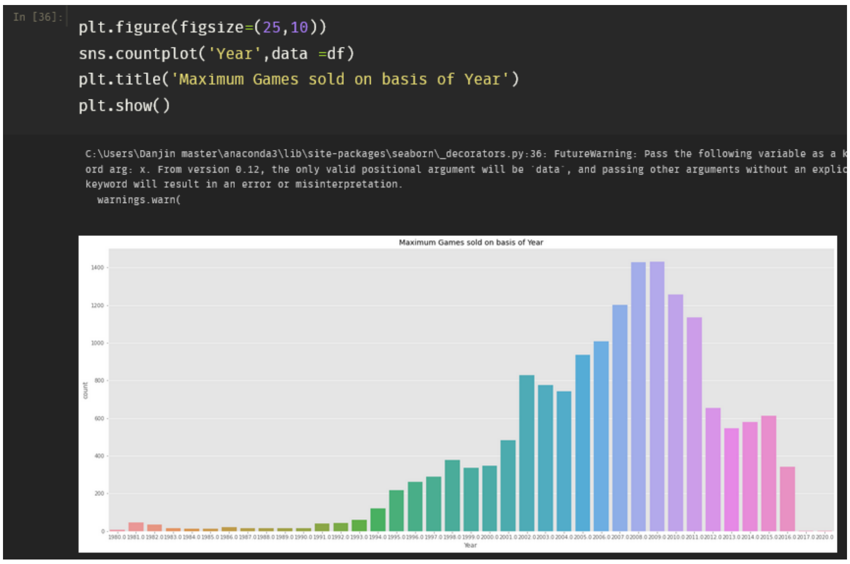




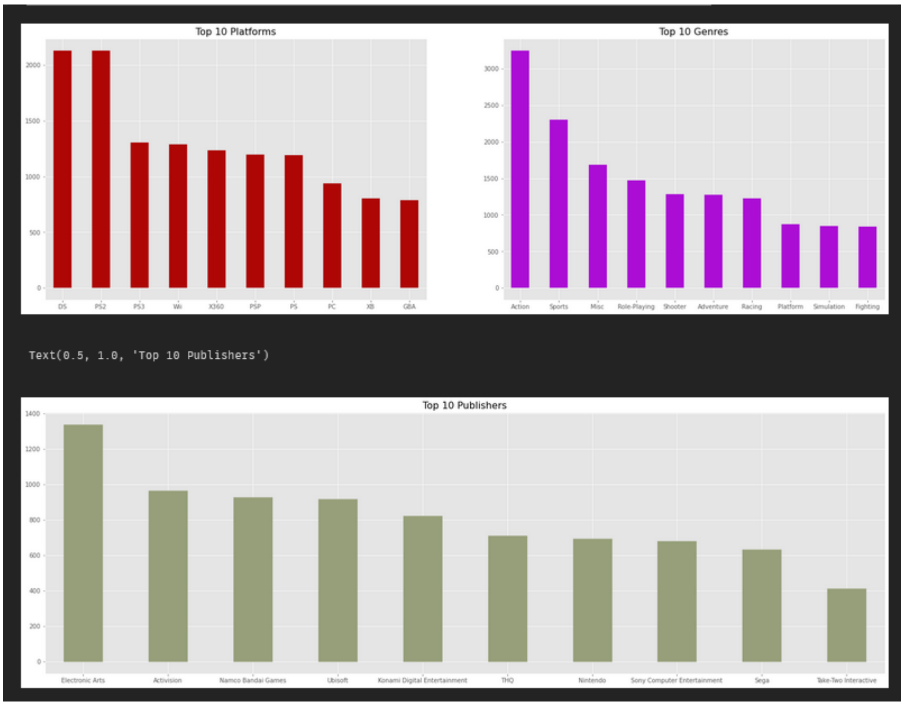


- The genre ‘Role-Playing’ has made more number sales in Japan.
- Whereas in North America and Europe, most sales were made by the genres ‘Shooter’ and ‘Platform’.
- In other regions and countries, the genres ‘Shooter’ and ‘Racing’ dominate the sales.

- Maximum game sold in the year of 2008- 2009.



- Top 10 platform, genre, and publishers:-

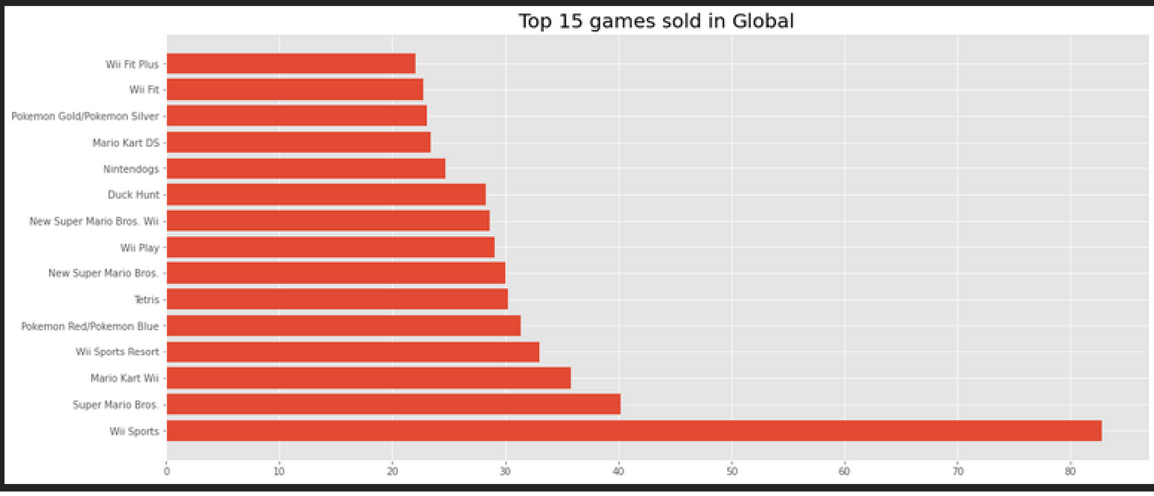


- ❖ DS and PS2 are the most popular platforms in comparison to other platforms.
- ❖ Action is the most popular genre and the second-most in the sports
- ❖ Electronic Arts have published 1300+ products

- See the correlation between sales:- Japanese sales are not correlated. Global and north sales are highly correlated.

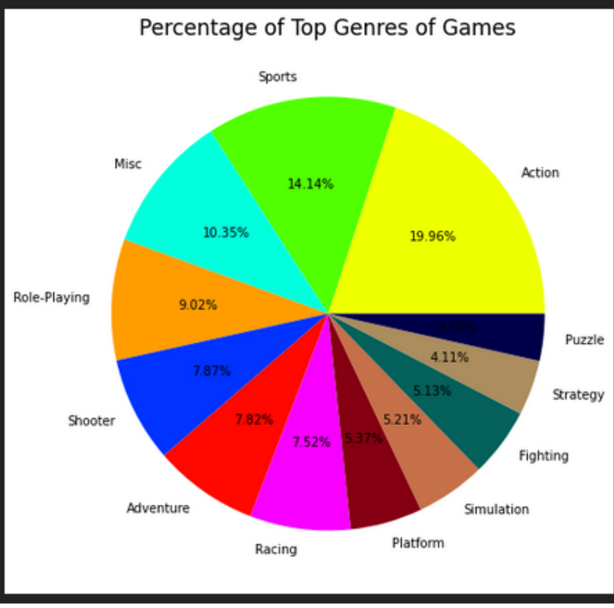
- Top 15 games sold globally.

```
top15 = df[0:15]
plt.figure(figsize = (18,8))
plt.barh(top15["Name"],top15["Global_Sales"], label = 'Top Games')
plt.title("Top 15 games sold in Global",fontdict = {"fontsize":20})
plt.show()
```



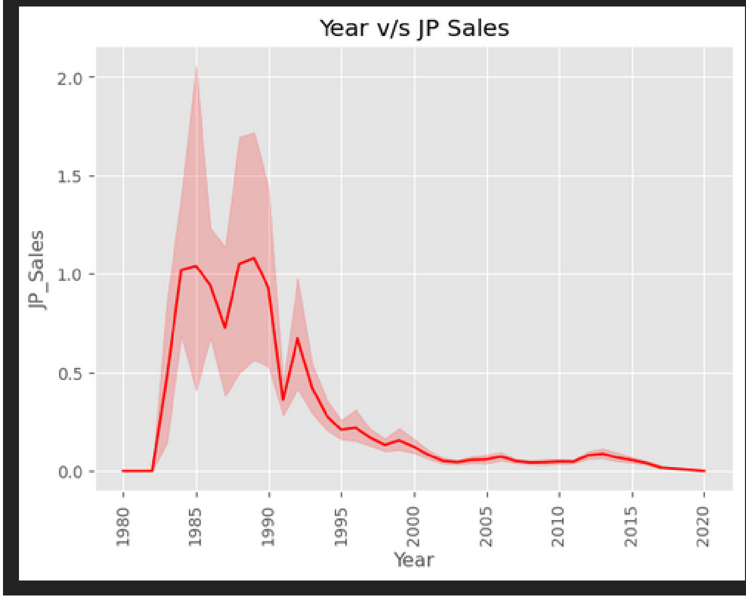
- Percentage of each genre of game.

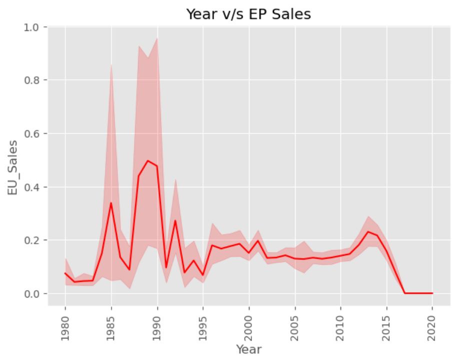
```
In [55]:
Genre = df.Genre
Genre = Genre.value_counts()
plt.figure(figsize = (8,8))
labels = Genre.index
colors = ["#eeff00","#51ff00","#00ffdd","#ff9d00","#0033ff","#ff0800","#f700ff","#850
plt.pie(Genre,labels = labels,colors = colors,autopct = "%.2f%%")
plt.title("Percentage of Top Genres of Games",fontdict = {"fontsize":17})
plt.savefig("Top Genres Chart",dpi = 200)
plt.show()
```

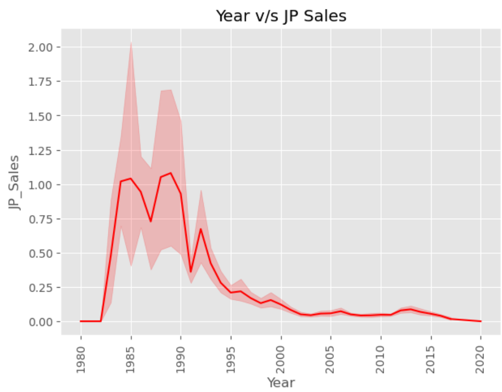


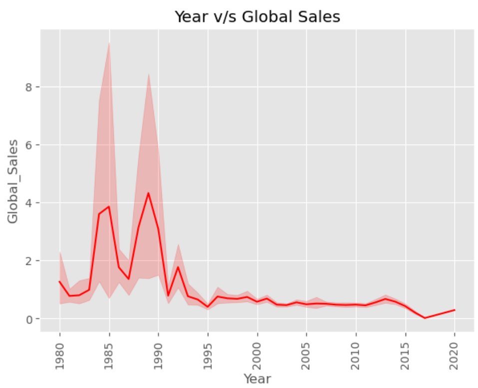
- Sales Graph.

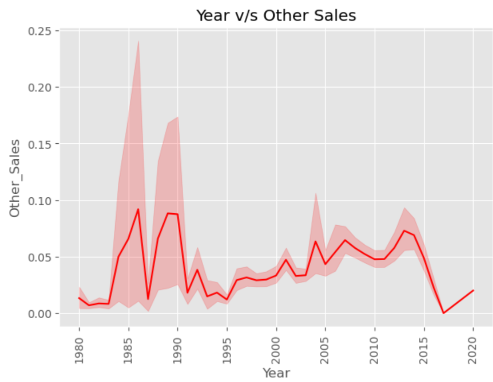
```python
# let's define a function to plot the graphs
def Year_plots(title, sales, color):
  plt.figure(figsize = (7,5), dpi= 100)
  plt.title(title)
  plt.xticks(rotation = 90)
  sns.lineplot(x = 'Year',
              y = sales,
              color = color,
              data = df)
Year_plots('Year v/s JP Sales', 'JP_Sales', 'red')
```



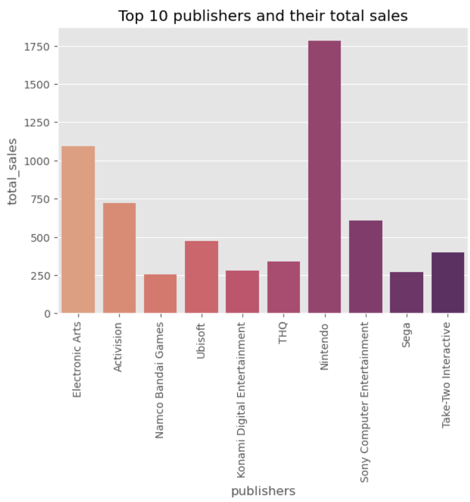


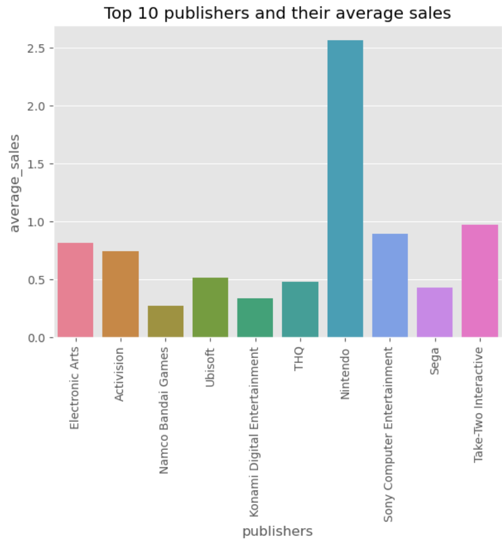






    - More sales in all the categories were made in the years 1984, 1985, 1988,1989,1990 and 1992
- Top publisher with their total sales and average sales.

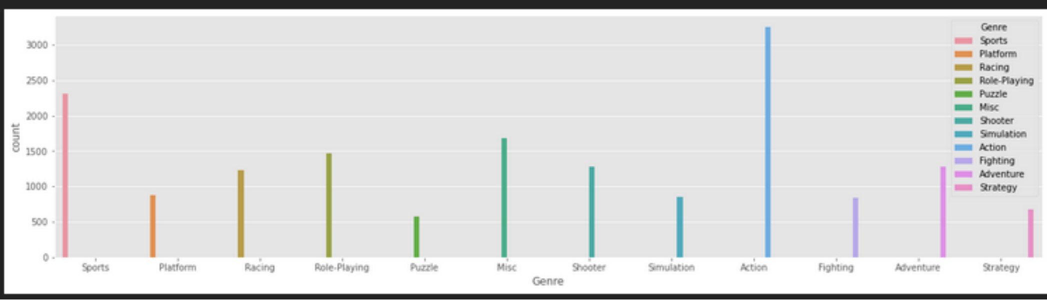




    - The company Nintendo has the most total sales and average sales as well.
- Check a number of genres.

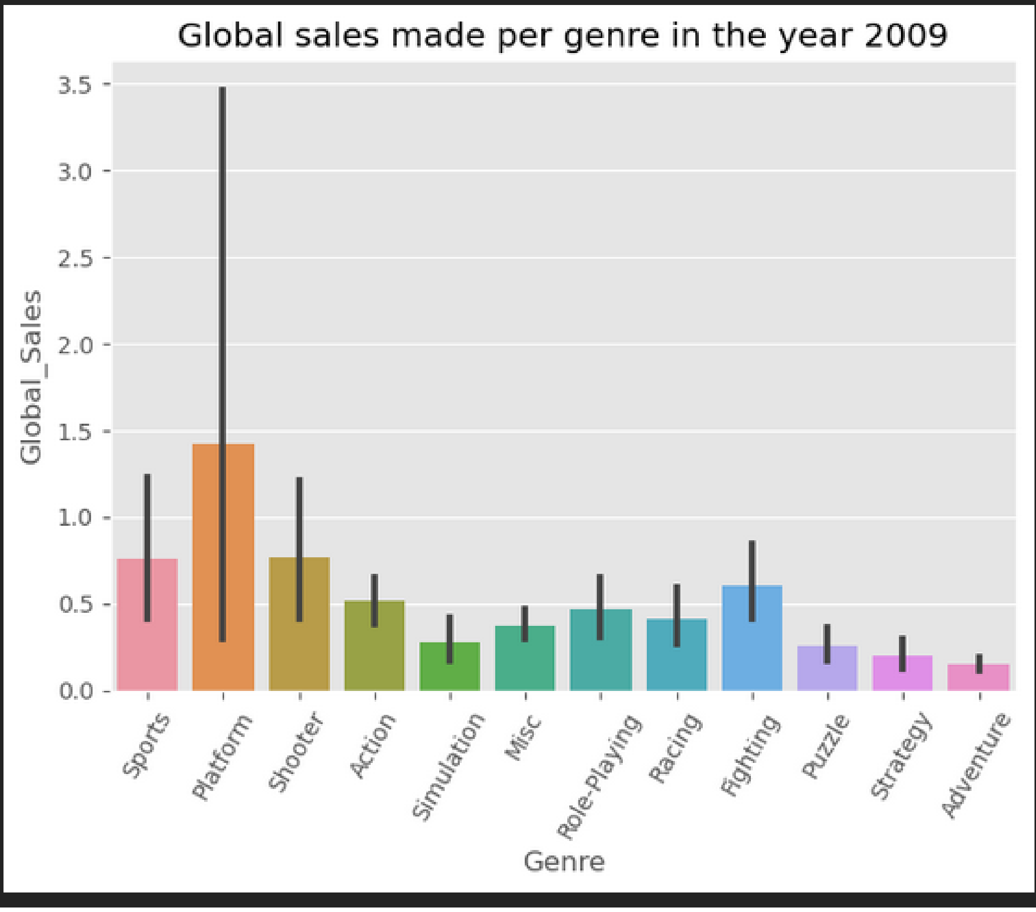
In [80]:
```
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt
plt.figure(figsize = (20,5))
sns.countplot('Genre', hue = 'Genre', data = df)
```

```
<AxesSubplot:xlabel='Genre', ylabel='count'>
```



- Global sales according to the genre in a particular year.

```
In [82]:
plt.figure(figsize= (7,5), dpi = 100)
plt.title('Global sales made per genre in the year 2009')
plt.xticks(rotation = 60)
sns.barplot(x = 'Genre',
            y = 'Global_Sales',
            data = year_2009_df);
```



- Electronic art publisher sold many games.

```
df['Publisher'].value_counts()[:10]
```

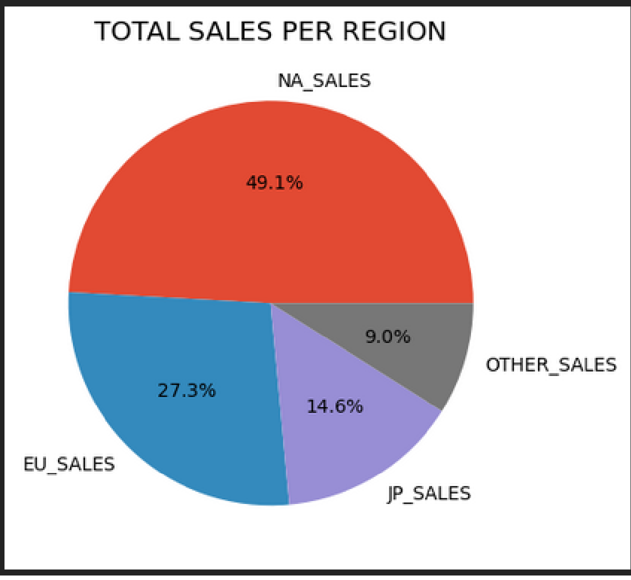
```
Electronic Arts                 1339
Activision                       966
Namco Bandai Games               928
Ubisoft                          918
Konami Digital Entertainment     823
THQ                              712
Nintendo                         696
Sony Computer Entertainment      682
Sega                             632
Take-Two Interactive             412
Name: Publisher, dtype: int64
```

- Check platform with all types of genre.

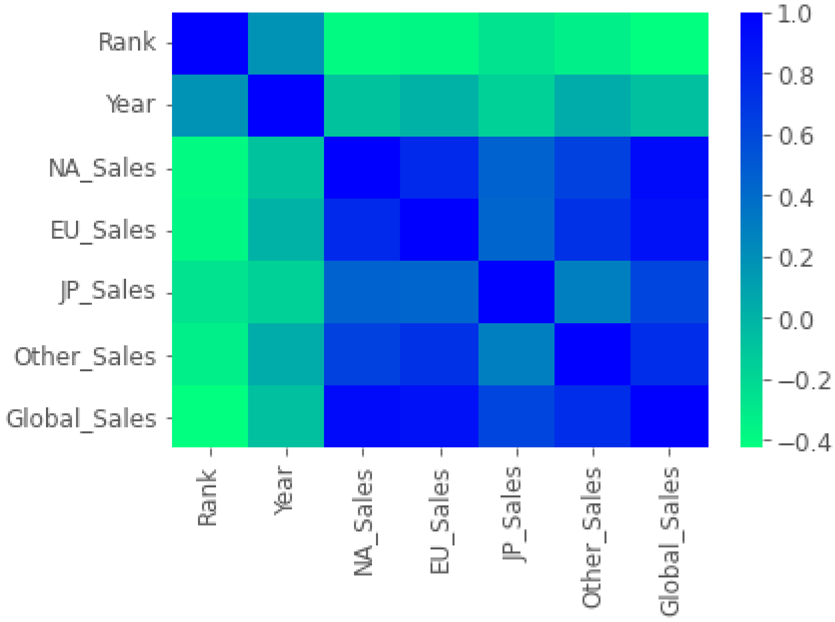
```
genre_table = pd.pivot_table(df,
                             index = 'Platform',
                             columns = ['Genre'],
                             values= ['NA_Sales','JP_Sales', 'EU_Sales',
                                      'Other_Sales', 'Global_Sales'],
                             aggfunc= np.sum)
genre_table.dropna(inplace = True)
# print(list(genre_table.index))
genre_table
```

- Highest total sales.

```
sales_cols = ['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales']
total = []
average = []
for sales_col in sales_cols:
  total.append(df[sales_col].sum())
  average.append(df[sales_col].mean())

plt.figure(figsize=(7,5), dpi = 100)
plt.title('Total Sales per region'.upper())
plt.pie(total, labels = [i.upper() for i in sales_cols],  autopct='%1.1f%%');
```

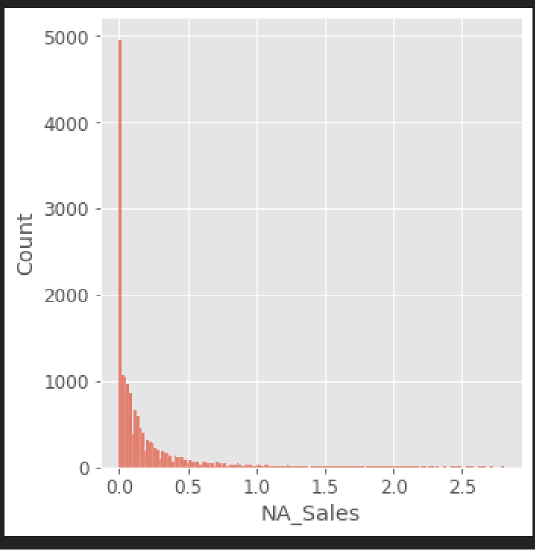


- Heatmap correlation.



- Checking Outliers.

```
sns.displot(data['NA_Sales'])
data2=data.copy()
q=data2['NA_Sales'].quantile(0.99)
data=data2[data2['NA_Sales']<q]
```



- Data Preprocessing.

```
data1=data.copy()
data1=pd.get_dummies(data,drop_first=True)
```

```
targets=data1['Global_Sales']
inputs=data1.drop(['Global_Sales'],axis=1)
```

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(inputs)
scaled_inputs=scaler.fit_transform(inputs)
```

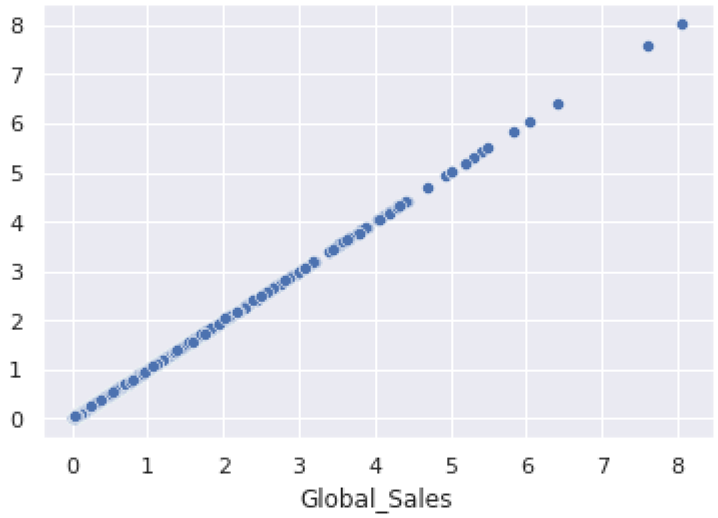
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(scaled_inputs,
                                               targets,test_size=0.2)
```

- Simple Model building and testing.

```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(x_train,y_train)
reg.score(x_train,y_train)
yhat=reg.predict(x_train)
```

```
reg.fit(x_test,y_test)
y_testhat=reg.predict(x_test)
reg.score(x_test,y_test)
```

```
sns.scatterplot(x=y_test,y=y_testhat)
```



- Model deployment:- For the user interface, we can deploy the model using streamlit, flask, and Django with a public server. Due to the large data or size of the project, we can't deploy this project on free services like Heroku. We can deploy on another time-limited free service like AWS, GCS, etc.

GITHUB LINK:- https://github.com/talkativetushar/VIDEO-games-sales-prediction

2. Recommendation for a new game or product.

In the gaming category, they are lots of games present in video game digital distribution like steam, rockstar. We can recommend the new game according to content-based or collaborative-based. We get the data from an online source. The data features maybe like the user, game, rating, how much playing category, etc. Using this we can recommend it. The pc component including processor, graphic card, etc can recommend by an e-commerce site. We take a dataset from Kaggle as well as assumption based we need the column of data for a content-based recommendation like game name, publisher, category/genre, description, etc. Text data to generate tags and similarity matrix. We make a content-based recommendation system. We also can make a collaborative-based recommendation system.

- **Market/Customer/Business needs Assessment:-** Our goal is to recommend a product to those potential customers in the near future. The customer buying preferences have been significantly changed due to the pandemic. Therefore, by using this technique, we aim to provide sales forecasting with useful insights from the available data and ways to generate more revenue.

- **Target Specification:-** The proposed system/service will provide the video game digital distribution with some techniques so that their sales boost up and they no longer have to go through an economic crisis.

- **External Search:-** Kaggle Competition
  https://www.kaggle.com/trolukovich/steam-games-complete-dataset

- **Applicable Patents:-** Data analysis tools of python and machine learning algorithms to predict sales in the future.

- **Applicable Regulations:-**
  - Data protection and privacy regulations(Customers)
  - Employment Laws
  - Regulations against false advertising.

- **Applicable Constraintsble:-** Continuous data collection and maintenance, Computer power according to data and focus on rarely bought products.

- **Business Opportunity:-** Since the above technique has only been used by large companies because of data but we can grow our own small business using all over market data analysis.

- **Concept Generation:-** Come from the recommendation system with some analysis.

- **Concept Development:-** Basically we need to predict sales in the future and Using analysis we can make a strategy to get more sales.

- **Final Product Prototype (abstract) with Schematic Diagram:-**
    - Collect data from Kaggle.
    - Analyzing.
    - Preprocessing.
    - Training.
    - Model creation & deployment.

GITHUB LINK:- https://github.com/talkativetushar/Game-recommendatio

- Import data.

In [1]:
```python
# from google.colab import drive
# drive.mount("/content/gdrive")
```

In [2]:
```python
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directc

games = pd.read_csv('steam_games.csv')
# games = pd.read_csv('/content/gdrive/My Drive/steam_games.csv')
games.head()

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you cre
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

	url	types	name	desc_snippet	recent_reviews	all_reviews	release_date	developer	publisher	popular_tags
0	https://store.steampowered.com /app/379720/DOOM/	app	DOOM	Now includes all three premium DLC packs (Unto...	Very Positive,(554),- 89% of the 554 user revi...	Very Positive,(42,550),- 92% of the 42,550 use...	May 12, 2016	id Software	Bethesda Softworks,Bethesda Softworks	FPS,Gore,Action,Demons,Shooter,First-Person,Gr...
1	https://store.steampowered.com /app/578080/PLAY...	app	PLAYERUNKNOWN'S BATTLEGROUNDS	PLAYERUNKNOWN'S BATTLEGROUNDS is a battle roya...	Mixed,(6,214),- 49% of the 6,214 user reviews ...	Mixed,(836,608),- 49% of the 836,608 user revi...	Dec 21, 2017	PUBG Corporation	PUBG Corporation,PUBG Corporation	Survival,Shooter,Multiplayer,Battle Royale,PvP...
2	https://store.steampowered.com /app/637090/BATT...	app	BATTLETECH	Take command of your own mercenary outfit of '...	Mixed,(166),- 54% of the 166 user reviews in t...	Mostly Positive,(7,030),- 71% of the 7,030 use...	Apr 24, 2018	Harebrained Schemes	Paradox Interactive,Paradox Interactive	Mechs,Strategy,Turn-Based,Turn-Based Tactics,S...
3	https://store.steampowered.com /app/221100/DayZ/	app	DayZ	The post-soviet country of Chernarus is struck...	Mixed,(932),- 57% of the 932 user reviews in t...	Mixed,(167,115),- 61% of the 167,115 user revi...	Dec 13, 2018	Bohemia Interactive	Bohemia Interactive,Bohemia Interactive	Survival,Zombies,Open World,Multiplayer,PvP,Ma...
4	https://store.steampowered.com /app/8500/EVE_On...	app	EVE Online	EVE Online is a community-driven spaceship MMO...	Mixed,(287),- 54% of the 287 user reviews in t...	Mostly Positive,(11,481),- 74% of the 11,481 u...	May 6, 2003	CCP	CCP,CCP	Space,Massively Multiplayer,Sci-fi,Sandbox,MMO...

- Check the shape and get only important text columns that are useful to make a recommendation system. We make a content-based recommendation system. We can make a collaborative based as well as. For content based we need tags and we need a behavior datasets for a collaborative-based recommendation model.

```
In [3]: games.shape
```

```
(40833, 20)
```

```
In [4]: #Get import column for data science
# dataGames = pd.read_csv('/content/gdrive/My Drive/steam_games.csv', usecols=["url","name", "genre", "game_deta
# dataGames.head()
dataGames = pd.read_csv('steam_games.csv', usecols=["url","name", "genre", "game_details", "popular_tags", "publ
dataGames.head()
```

	url	name	developer	publisher	popular_tags	game_details	genre
0	https://store.steampowered.com /app/379720/DOOM/	DOOM	id Software	Bethesda Softworks,Bethesda Softworks	FPS,Gore,Action,Demons,Shooter,First-Person,Gr...	Single-player,Multi-player,Co-op,Steam Achieve...	Action
1	https://store.steampowered.com /app/578080/PLAY...	PLAYERUNKNOWN'S BATTLEGROUNDS	PUBG Corporation	PUBG Corporation,PUBG Corporation	Survival,Shooter,Multiplayer,Battle Royale,PvP...	Multi-player,Online Multi-Player,Stats	Action,Adventure,Massively Multiplayer
2	https://store.steampowered.com /app/637090/BATT...	BATTLETECH	Harebrained Schemes	Paradox Interactive,Paradox Interactive	Mechs,Strategy,Turn-Based,Turn-Based Tactics,S...	Single-player,Multi-player,Online Multi-Player...	Action,Adventure,Strategy
3	https://store.steampowered.com /app/221100/DayZ/	DayZ	Bohemia Interactive	Bohemia Interactive,Bohemia Interactive	Survival,Zombies,Open World,Multiplayer,PvP,Ma...	Multi-player,Online Multi-Player,Steam Worksho...	Action,Adventure,Massively Multiplayer
4	https://store.steampowered.com /app/8500/EVE_On...	EVE Online	CCP	CCP,CCP	Space,Massively Multiplayer,Sci-fi,Sandbox,MMO...	Multi-player,Online Multi-Player,MMO,Co-op,Onl...	Action,Free to Play,Massively Multiplayer,RPG,...

- Check null values and handle them.

```
In [7]: withoutnullGames = dataGames.dropna(how='any',axis=0)
withoutnullGames.head()
```

	url	name	developer	publisher	popular_tags	game_details	genre
0	https://store.steampowered.com /app/379720/DOOM/	DOOM	id Software	Bethesda Softworks,Bethesda Softworks	FPS,Gore,Action,Demons,Shooter,First-Person,Gr...	Single-player,Multi-player,Co-op,Steam Achieve...	Action
1	https://store.steampowered.com /app/578080/PLAY...	PLAYERUNKNOWN'S BATTLEGROUNDS	PUBG Corporation	PUBG Corporation,PUBG Corporation	Survival,Shooter,Multiplayer,Battle Royale,PvP...	Multi-player,Online Multi-Player,Stats	Action,Adventure,Massively Multiplayer
2	https://store.steampowered.com /app/637090/BATT...	BATTLETECH	Harebrained Schemes	Paradox Interactive,Paradox Interactive	Mechs,Strategy,Turn-Based,Turn-Based Tactics,S...	Single-player,Multi-player,Online Multi-Player...	Action,Adventure,Strategy
3	https://store.steampowered.com /app/221100/DayZ/	DayZ	Bohemia Interactive	Bohemia Interactive,Bohemia Interactive	Survival,Zombies,Open World,Multiplayer,PvP,Ma...	Multi-player,Online Multi-Player,Steam Worksho...	Action,Adventure,Massively Multiplayer
4	https://store.steampowered.com /app/8500/EVE_On...	EVE Online	CCP	CCP,CCP	Space,Massively Multiplayer,Sci-fi,Sandbox,MMO...	Multi-player,Online Multi-Player,MMO,Co-op,Onl...	Action,Free to Play,Massively Multiplayer,RPG,...

- Data preprocessing for tags.

```
In [15]: import re
for i, row in withoutnullGames.iterrows():
    clean = re.sub('[^A-Za-z0-9]+', '', row["name"])
    clean = clean.lower()
    dataGames.at[i, 'ID'] = clean
```

```
In [16]: withoutnullGames.head()
```

	url	name	developer	publisher	popular_tags	game_details	genre
0	https://store.steampowered.com /app/379720/DOOM/	DOOM	id Software	Bethesda Softworks,Bethesda Softworks	FPS,Gore,Action,Demons,Shooter,First-Person,Gr...	Single-player,Multi-player,Co-op,Steam Achieve...	Action
1	https://store.steampowered.com /app/578080/PLAY...	PLAYERUNKNOWN'S BATTLEGROUNDS	PUBG Corporation	PUBG Corporation,PUBG Corporation	Survival,Shooter,Multiplayer,Battle Royale,PvP...	Multi-player,Online Multi-Player,Stats	Action,Adventure,Massively Multiplayer
2	https://store.steampowered.com /app/637090/BATT...	BATTLETECH	Harebrained Schemes	Paradox Interactive,Paradox Interactive	Mechs,Strategy,Turn-Based,Turn-Based Tactics,S...	Single-player,Multi-player,Online Multi-Player...	Action,Adventure,Strategy
3	https://store.steampowered.com /app/221100/DayZ/	DayZ	Bohemia Interactive	Bohemia Interactive,Bohemia Interactive	Survival,Zombies,Open World,Multiplayer,PvP,Ma...	Multi-player,Online Multi-Player,Steam Worksho...	Action,Adventure,Massively Multiplayer
4	https://store.steampowered.com /app/8500/EVE_On...	EVE Online	CCP	CCP,CCP	Space,Massively Multiplayer,Sci-fi,Sandbox,MMO...	Multi-player,Online Multi-Player,MMO,Co-op,Onl...	Action,Free to Play,Massively Multiplayer,RPG,...

```
In [17]: def clean_data(x):
    if isinstance(x, str):
        return x.replace(" ", "")
    else:
        print(x)
        return x
```

- Make a Tags column.

```
withoutnullGames['tags'] = withoutnullGames['developer'] +
withoutnullGames['publisher'] + withoutnullGames['popular_tags'] + withoutnullGames['game_details']
+ withoutnullGames['genre']
```

- Use Stemming and Vectorization.

```
import nltk
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
def stem(text):
  y = []
  for i in text.split():
    y.append(ps.stem(i))

  return " ".join(y)

new['tags'] = new['tags'].apply(stem)
```

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=5000,stop_words='english')
```

```
vector = cv.fit_transform(new['tags']).toarray()
```

- Make similarity matrix and recommendation function.

```
from sklearn.metrics.pairwise import cosine_similarity
similarity = cosine_similarity(vector)
```

```
similarity[0]
```

```
array([1.        , 0.34006802, 0.44394139, ..., 0.29643243, 0.4631392 ,
       0.18077538])
```

```
similarity[1]
```

```
array([0.34006802, 1.        , 0.30733234, ..., 0.30242157, 0.04973647,
       0.11065667])
```

```
def recommend(game):
    index = new[new['name'] == game].index[0]
    distances = sorted(list(enumerate(similarity[index])),reverse=True,key = lambda x: x[1])
    for i in distances[1:6]:
        print(new.iloc[i[0]][1])
```

- Testing.

```
recommend('DayZ')
```

```
Unturned
CASE 2: Animatronics Survival
Miscreated
Tower Unite
Rust
```

- Deployment:- Using streamlit, flask, and Django, we can convert this model into a user interface and deploy it on the server. Need more computational power to make a model if the data is big because we save the similarity matrix for deployment and the similarity matrix is based on users x users.

3. Spam and recommendation reviews classification.

A lot of gamers write reviews on the game page and have the option of choosing whether they would recommend this game to others or not. However, determining this sentiment automatically from the text can help Steam to automatically tag such reviews extracted from other forums across the internet and can help them better judge the popularity of games. Predict whether the reviewer recommended the game titles available in the test set on the basis of review text and other information.

- **Market/Customer/Business needs Assessment:-** There are many games of play store are clickbait like the developer add the category game name as already game but in this only wallpaper. So the user gives him a bad review. So using a machine learning algorithm, we can detect the game's review is good or not. If they are not good. We will not recommend this game. This is one problem of review but there are lots of problems. This model can be useful for game digital distribution services, play stores, etc.

- **Target Specification:-** The proposed system/service will provide the video game digital distribution, play store, etc. with some techniques so that their sales boost up using reviews classification and they no longer have to go through an economic crisis.

- **External Search:-** Kaggle Competition Kaggle:- https://www.kaggle.com/arashnic/game-review-dataset

- **Benchmarking:-** Video game digital distribution giants like Steam, rockstar have been using affinity analysis to perform Market Basket Analysis, which identifies purchasing habits of customers and uses this information to cross-sell and up-sell relevant items.

- **Applicable Constraintsble:-** Continuous data collection and maintenance, Computer power according to data and focus on rarely bought products.

- **Business Opportunity:-** Since the above technique has only been used by large companies because of data but we can grow our own small business using all over market data analysis.

- **Concept Generation:-** Come from the recommendation system with some analysis.

- **Concept Development:-** Basically we need to predict sales in the future and Using analysis we can make a strategy to get more sales.

- **Final Product Prototype (abstract) with Schematic Diagram:-**
    - Collect data from Kaggle.
    - Analyzing.
    - Preprocessing.
    - Training.
    - Model creation & deployment.

GITHUB LINK:- https://github.com/talkativetushar/Recommended-review-spam-classification

---

- Import data

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('ggplot')
# df = pd.read_csv('train.csv')
df = pd.read_csv('/content/gdrive/My Drive/train.csv')
df.head()
```

	review_id	title	year	user_review	user_suggestion
0	1	Spooky's Jump Scare Mansion	2016.0	I'm scared and hearing creepy voices. So I'll...	1
1	2	Spooky's Jump Scare Mansion	2016.0	Best game, more better than Sam Pepper's YouTu...	1
2	3	Spooky's Jump Scare Mansion	2016.0	A littly iffy on the controls, but once you kn...	1
3	4	Spooky's Jump Scare Mansion	2015.0	Great game, fun and colorful and all that.A si...	1
4	5	Spooky's Jump Scare Mansion	2015.0	Not many games have the cute tag right next to...	1

- Basic Operation.

```
[6] df.drop(columns=['review_id','year','title'],inplace=True)
```

-

```
[10] df.duplicated().sum()

3

[11] # remove duplicates
     df = df.drop_duplicates(keep='first')

[12] df.duplicated().sum()

0
```

- Feature enginnering.

```
[18] df['num_words'] = df['user_review'].apply(lambda x:len(nltk.word_tokenize(x)))
     df['num_characters'] = df['user_review'].apply(len)
     df['num_sentences'] = df['user_review'].apply(lambda x:len(nltk.sent_tokenize(x)))
     df.head()
```

	user_review	user_suggestion	num_characters	num_words	num_sentences
0	I'm scared and hearing creepy voices. So I'll...	1	710	152	5
1	Best game, more better than Sam Pepper's YouTu...	1	335	61	3
2	A littly iffy on the controls, but once you kn...	1	397	84	5
3	Great game, fun and colorful and all that.A si...	1	280	62	6
4	Not many games have the cute tag right next to...	1	334	72	4

```
# not recommended
df[df['user_suggestion'] == 0][['num_characters','num_words','num_sentences']].describe()
```

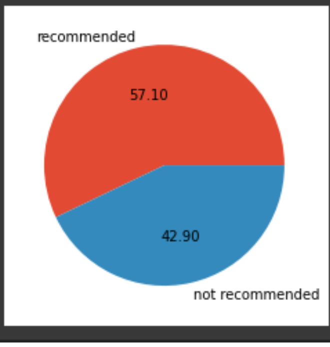
	num_characters	num_words	num_sentences
count	7427.000000	7427.000000	7427.000000
mean	800.724653	164.995153	6.558368
std	944.470198	193.828364	7.764420
min	6.000000	1.000000	1.000000
25%	285.000000	59.000000	2.000000
50%	473.000000	98.000000	4.000000
75%	891.000000	185.000000	8.000000
max	7989.000000	1711.000000	119.000000

```
[21] #recommended
     df[df['user_suggestion'] == 1][['num_characters','num_words','num_sentences']].describe()
```

	num_characters	num_words	num_sentences
count	9886.000000	9886.000000	9886.000000
mean	730.524277	150.165385	6.116124
std	866.760529	178.849629	7.691528
min	6.000000	1.000000	1.000000
25%	267.000000	55.000000	2.000000
50%	434.000000	90.000000	4.000000
75%	820.000000	170.000000	7.000000
max	8000.000000	2861.000000	196.000000

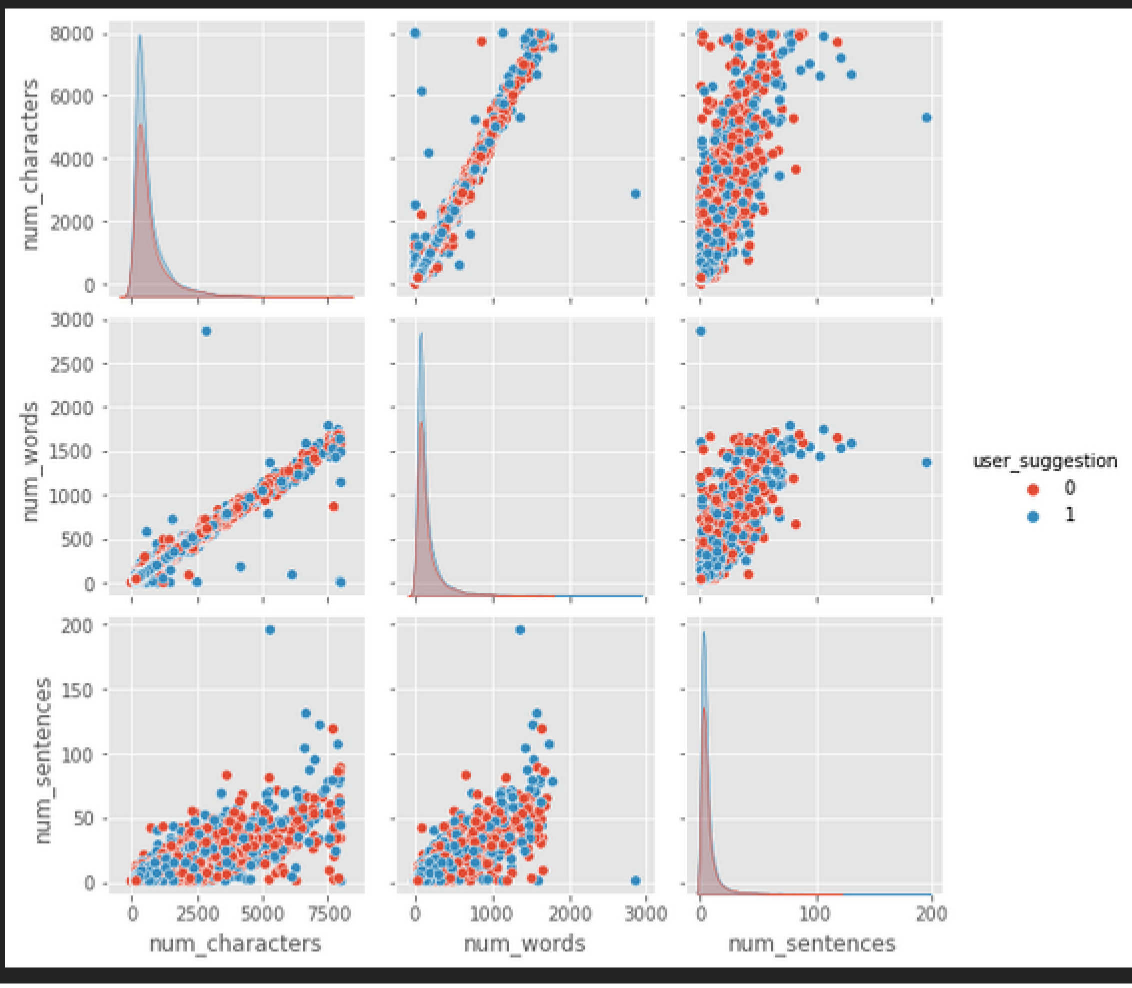
- Some analysis

```
import matplotlib.pyplot as plt
plt.pie(df['user_suggestion'].value_counts(), labels=['recommended','not recommended'],autopct="%0.2f")
plt.show()
```




```
sns.pairplot(df,hue='user_suggestion')
```

```
<seaborn.axisgrid.PairGrid at 0x18d659cdc40>
```

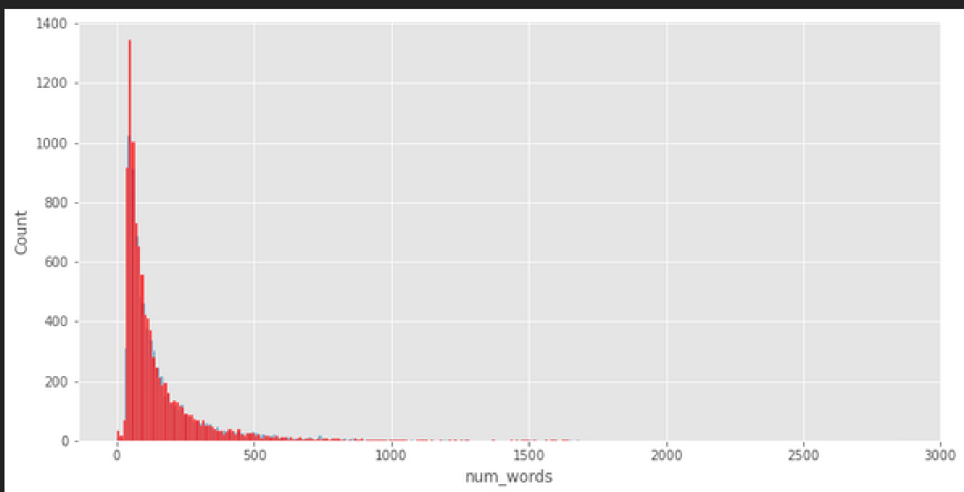


```
In [27]: sns.heatmap(df.corr(),annot=True)
```

```
<AxesSubplot:>
```


```
In [25]: plt.figure(figsize=(12,6))
         sns.histplot(df[df['user_suggestion'] == 0]['num_words'])
         sns.histplot(df[df['user_suggestion'] == 1]['num_words'],color='red')
```

```
<AxesSubplot:xlabel='num_words', ylabel='Count'>
```



- Data preprocessing.

```
In [48]:
# Data Preprocessing
# Lower case, Tokenization, Removing special characters, Removing stop words and punctuation,
from nltk.corpus import stopwords
nltk.download('stopwords')
def transform_text(text):
    text = text.lower()
    text = nltk.word_tokenize(text)
    y = []
    for i in text:
        if i.isalnum():
            y.append(i)
    text = y[:]
    y.clear()
    for i in text:
        if i not in stopwords.words('english'):
            y.append(i)
    text = y[:]
    y.clear()
    for i in text:
        y.append(ps.stem(i))
    return " ".join(y)
```

```
In [50]:
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
```

```
In [51]:
df['transformed_text'] = df['user_review'].apply(transform_text)
```

```
In [52]:
df.head()
```

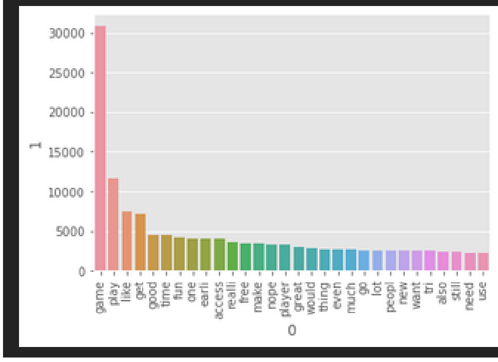
	user_review	user_suggestion	num_characters	num_words	num_sentences	transformed_text
0	I'm scared and hearing creepy voices. So I'll...	1	710	154	5	scare hear creepi voic paus moment write revie...
1	Best game, more better than Sam Pepper's YouTu...	1	335	61	3	best game better sam pepper youtub account nee...
2	A littly iffy on the controls, but once you kn...	1	397	84	5	littli iffi control know play easi master made...
3	Great game, fun and colorful and all that.A si...	1	280	61	6	great game fun color side note though get wind...
4	Not many games have the cute tag right next to...	1	334	72	4	mani game cute tag right next horror tag first...

- Analysis after preprocessing.

```
spam_corpus = []
for msg in df[df['user_suggestion'] == 1]['transformed_text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)
```

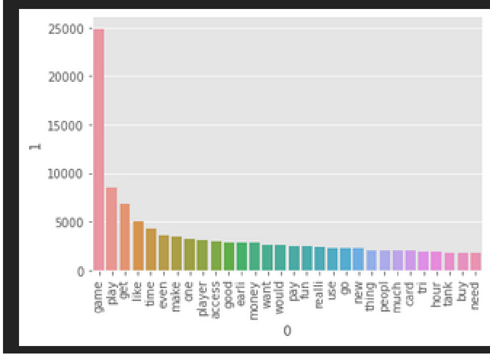
```
from collections import Counter
sns.barplot(pd.DataFrame(Counter(spam_corpus).most_common(30))[0],pd.DataFrame(Counter(spam_corpu
plt.xticks(rotation='vertical')
plt.show()
```

```
C:\Users\Danjin master\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. Fr
om version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error
or misinterpretation.
  warnings.warn(
```




```
ham_corpus = []
for msg in df[df['user_suggestion'] == 0]['transformed_text'].tolist():
    for word in msg.split():
        ham_corpus.append(word)


from collections import Counter
sns.barplot(pd.DataFrame(Counter(ham_corpus).most_common(30))[0],pd.DataFrame(Counter(ham_corpus)
plt.xticks(rotation='vertical')
plt.show()
```

```
C:\Users\Danjin master\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. Fr
om version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error
or misinterpretation.
  warnings.warn(
```



- Model Building

```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
```

```
X = tfidf.fit_transform(df['transformed_text']).toarray()
```

```
y = df['user_suggestion'].values
```

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

```
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score
```

```
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
```

```
gnb.fit(X_train,y_train)
y_pred1 = gnb.predict(X_test)
print(accuracy_score(y_test,y_pred1))
print(confusion_matrix(y_test,y_pred1))
print(precision_score(y_test,y_pred1))
```

```
0.7585908172106111
[[1116  368]
 [ 468 1511]]
0.8041511442256519
```

```
mnb.fit(X_train,y_train)
y_pred2 = mnb.predict(X_test)
print(accuracy_score(y_test,y_pred2))
print(confusion_matrix(y_test,y_pred2))
print(precision_score(y_test,y_pred2))
```

```
0.8206757146982385
[[1082  402]
 [ 219 1760]]
0.8140610545790934
```

```
bnb.fit(X_train,y_train)
y_pred3 = bnb.predict(X_test)
print(accuracy_score(y_test,y_pred3))
print(confusion_matrix(y_test,y_pred3))
print(precision_score(y_test,y_pred3))
```

```
0.7877562806814901
[[ 993  491]
 [ 244 1735]]
0.7794249775381851
```

```python
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
```

```python
svc = SVC(kernel='sigmoid', gamma=1.0)
knc = KNeighborsClassifier()
mnb = MultinomialNB()
dtc = DecisionTreeClassifier(max_depth=5)
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=50, random_state=2)
abc = AdaBoostClassifier(n_estimators=50, random_state=2)
bc = BaggingClassifier(n_estimators=50, random_state=2)
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)
gbdt = GradientBoostingClassifier(n_estimators=50,random_state=2)
xgb = XGBClassifier(n_estimators=50,random_state=2)
```

```python
clfs = {
    'SVC' : svc,
    'KN' : knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'AdaBoost': abc,
    'BgC': bc,
    'ETC': etc,
    'GBDT':gbdt,
    'xgb':xgb
}
```

```python
def train_classifier(clf,X_train,y_train,X_test,y_test):
    clf.fit(X_train,y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test,y_pred)
    precision = precision_score(y_test,y_pred)

    return accuracy,precision

train_classifier(svc,X_train,y_train,X_test,y_test)
```

```
accuracy_scores = []
precision_scores = []

for name,clf in clfs.items():

    current_accuracy,current_precision = train_classifier(clf, X_train,y_train,X_test,y_test)

    print("For ",name)
    print("Accuracy - ",current_accuracy)
    print("Precision - ",current_precision)

    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)
```

```
performance_df = pd.DataFrame({'Algorithm':clfs.keys(),
                               'Accuracy':accuracy_scores,
                               'Precision':precision_scores}).sort_values('Precision',ascending=
```

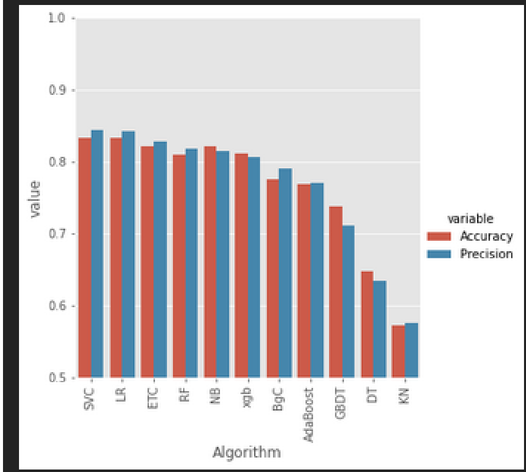
```
performance_df
```

	Algorithm	Accuracy	Precision
0	SVC	0.833381	0.843627
4	LR	0.832515	0.842054
8	ETC	0.820676	0.828337
5	RF	0.809703	0.817919
2	NB	0.820676	0.814061
10	xgb	0.811146	0.806288
7	BgC	0.775051	0.789296
6	AdaBoost	0.768698	0.769689
9	GBDT	0.736645	0.710620
3	DT	0.647704	0.634052
1	KN	0.572336	0.574151

```
performance_df1 = pd.melt(performance_df, id_vars = "Algorithm")
performance_df1
```

	Algorithm	variable	value
0	SVC	Accuracy	0.833381
1	LR	Accuracy	0.832515
2	ETC	Accuracy	0.820676
3	RF	Accuracy	0.809703
4	NB	Accuracy	0.820676
5	xgb	Accuracy	0.811146
6	BgC	Accuracy	0.775051
7	AdaBoost	Accuracy	0.768698
8	GBDT	Accuracy	0.736645
9	DT	Accuracy	0.647704
10	KN	Accuracy	0.572336
11	SVC	Precision	0.843627
12	LR	Precision	0.842054
13	ETC	Precision	0.828337
14	RF	Precision	0.817919
15	NB	Precision	0.814061
16	xgb	Precision	0.806288
17	BgC	Precision	0.789296
18	AdaBoost	Precision	0.769689
19	GBDT	Precision	0.710620
20	DT	Precision	0.634052
21	KN	Precision	0.574151

```
sns.catplot(x = 'Algorithm', y='value',
               hue = 'variable',data=performance_df1, kind='bar',height=5)
plt.ylim(0.5,1.0)
plt.xticks(rotation='vertical')
plt.show()
```



```python
# Voting Classifier
svc = SVC(kernel='sigmoid', gamma=1.0,probability=True)
mnb = MultinomialNB()
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)

from sklearn.ensemble import VotingClassifier

voting = VotingClassifier(estimators=[('svm', svc), ('nb', mnb), ('et', etc)],voting='soft')

voting.fit(X_train,y_train)
```

```
VotingClassifier(estimators=[('svm',
                              SVC(gamma=1.0, kernel='sigmoid',
                                  probability=True)),
                             ('nb', MultinomialNB()),
                             ('et',
                              ExtraTreesClassifier(n_estimators=50,
                                                   random_state=2))],
                 voting='soft')
```

```python
y_pred = voting.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))
```

```
Accuracy 0.8414669361825807
Precision 0.8404761904761905
```

```python
# Applying stacking
estimators=[('svm', svc), ('nb', mnb), ('et', etc)]
final_estimator=RandomForestClassifier()


from sklearn.ensemble import StackingClassifier

clf = StackingClassifier(estimators=estimators, final_estimator=final_estimator)

clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))
```

```python
import pickle
pickle.dump(tfidf,open('vectorizer.pkl','wb'))
pickle.dump(voting,open('model.pkl','wb'))
```

- Deployment:- Using streamlit, flask, and Django, we can convert this model into a user interface and deploy it on the server. Need more computational power with time to make a model if the data is big.

4. Creating bots and AI games.

AI can make a gaming bot that looks like a real player using reinforcement learning and they help users for practicing or in a game. This problem needs domain knowledge with machine learning type reinforcement learning.

Credit:-
https://www.freecodecamp.org/news/how-to-build-an-ai-game-bot-using-openai-gym-and-universe-f2eb9bfbb40a/

5. Anti-cheat detection.

Robust Vision-Based Cheat Detection in Competitive Gaming is help for user or gaming experience. This task needs to know more about domain knowledge like game development.

Research paper:-
https://openworks.wooster.edu/cgi/viewcontent.cgi?article=11803&context=independentstudy

6. Market and gamer category segmentation.

Using segmentation of gamers, we can recommender relevant games to a person.
Credit:- https://www.breakingthewheel.com/video-game-market-segmentation/

7. Fighting Hate Speech and Trolls classification(Analysing chat in the game)

The all-online games have features that both team players can talk to each other with text or speak. There is so much bully communication happening. So using ai we can detect the person who talks to the bully then we can do it punishment like a ban. It's the same as spam reviews detection classification.

Credit:- https://paperswithcode.com/task/hate-speech-detection

8. Make a cluster in an open-world game.

There are lots of open-world survival online games like PUBG, super people, Fortnite, apex, etc. The zone is created like a cluster by some time for the duration of the match. This problem needs domain knowledge and we make a cluster using unsupervised learning with its algorithm.

9. To get an overview of Self-driving cars.

Using this we can get experience and overview to make a self-driving car in the virtual world. Let's code for a simple self-driving car.

CODE:- https://github.com/talkativetushar/Self-driving-car
Refrence:- https://github.com/SullyChen/Autopilot-TensorFlow

10. Sensitivity Finder.

Most gamers have problems finding the best sensitivity in online shooting games. Data science can solve it using some mathematical tools. The online shooting games have a shooting or training ground that includes most of the practice exercises using cardboard bots. Let's take an example of a valorant game, In the training ground, they have many tasks to improve the aim like 30 bots flickering rapidly, etc. so we make a dataset which includes attempt, sensitivity, dpi, gun, speed mode, how many bots hitting in a particular time, accuracy, etc. Using resampling we can find the best sensitivity.

This technique can be used in aiming software like aim lab, Kovak, etc. They can get the data using user performance and find the best sensitivity of the user.

	A	B	C	D	E	F	G	H	I
1	**Attempt Number**	**Date**	**Gun**	**DPI**	**Speed**	**Strafe**	**Sensitivity**	**Score**	**Accuracy**
2	1	12/05/2020	Vandal	1800	Medium	Off	0.80	8	27%
3	2	12/05/2020	Vandal	1800	Medium	Off	0.70	15	50%
4	3	12/05/2020	Vandal	1800	Medium	Off	0.70	15	50%
5	4	12/05/2020	Vandal	1800	Medium	Off	0.70	15	50%
6	5	12/05/2020	Vandal	1800	Medium	Off	0.65	18	60%
7	6	12/05/2020	Vandal	1800	Medium	Off	0.65	19	63%
8	7	12/05/2020	Vandal	1800	Medium	Off	0.65	15	50%

Reference :-
https://medium.com/@pinata.dev/how-data-science-saved-my-valorant-accuracy-58d1238a951

Alternative sensitivity finder by using ai is make an ai product of specific mouse and they calculate the movement of a mouse using hand then predict the sensitivity.