SC2 Team 7: Bryan Lee, Chow Wei Jie, Ang Yu Juan

SC1015 MINI PROJECT

CONTENT

1

Step 1

Data Preparation

2

Step 2

Exploratory Data Analysis

3

Step 3

Model Creation

4

Step 4

Conclusion

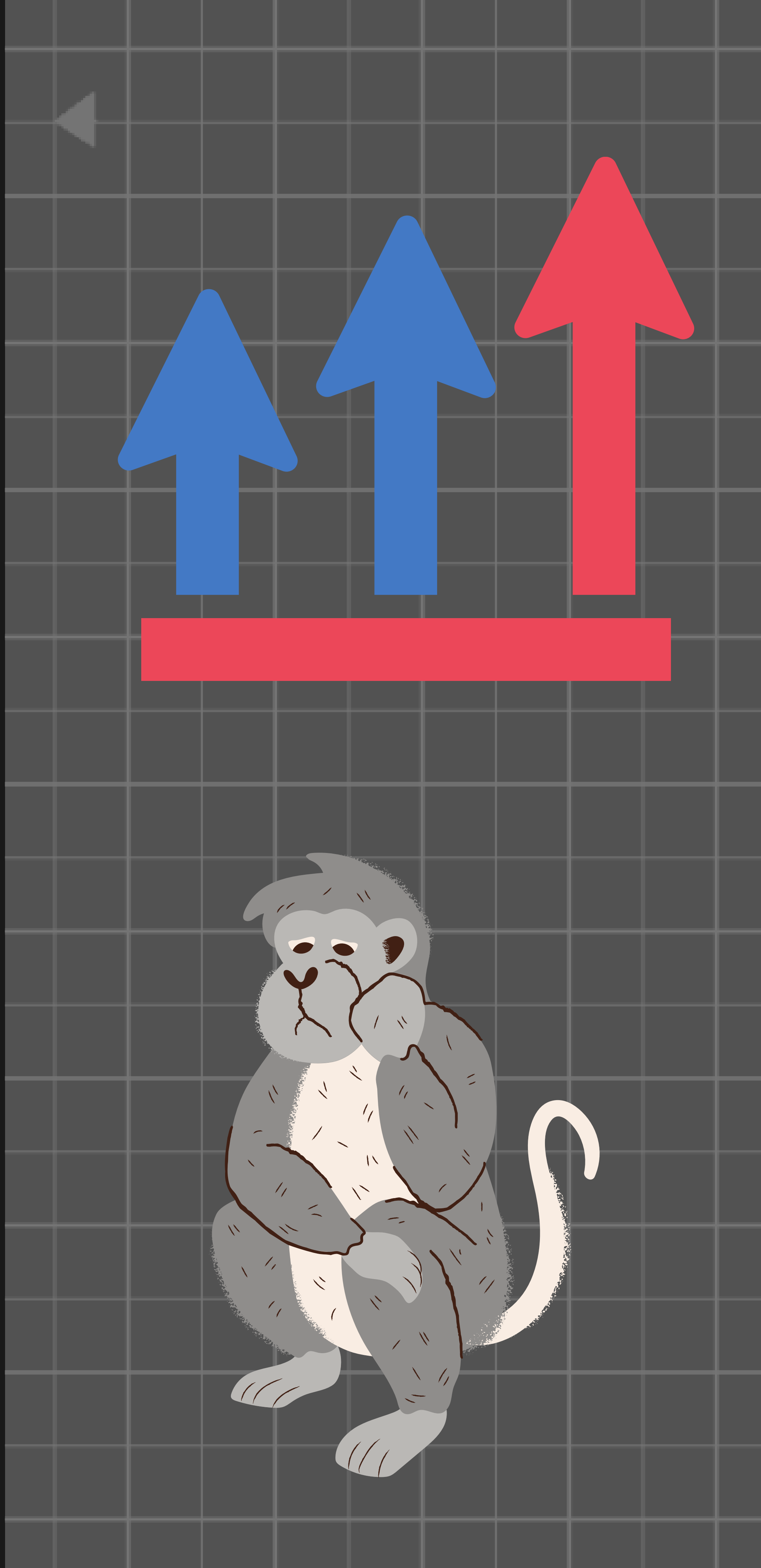
MOTIVATION

- More than 10,000 games are released last year

- Booming game industry

- Games are costly

- Increasingly difficult to find the 'right' game



PROBLEM STATEMENT

- How can we tell whether a game is good?

- How do we find games most suited for us?

DATASET

Steam

Background

Video game digital distribution service and storefront by Valve

Popularity

More than 50,000 games under Steam

OBTAINING THE DATASET

- Kaggle Steam Data (2 CSV)

  https://www.kaggle.com/datasets/nikdavis/steam-store-raw

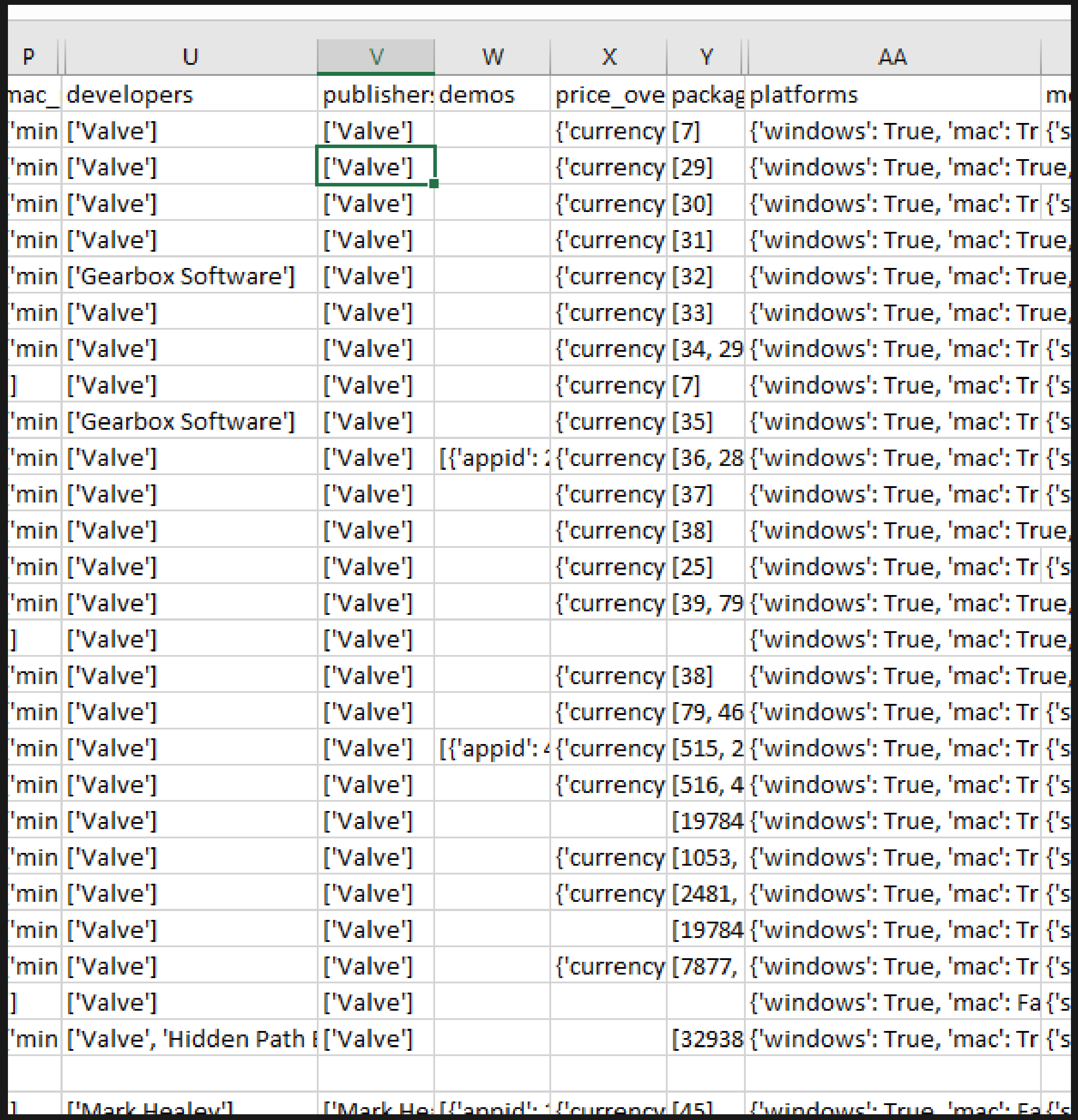
- Steam User Dataset (SQL)

  https://steam.internet.byu.edu/

- Steam API

  https://steamcommunity.com/dev

CLEANING THE DATASET



P	U	V	W	X	Y	AA	
mac_	developers	publishers	demos	price_ove	packag	platforms	m
'min	['Valve']	['Valve']		{'currency	[7]	{'windows': True, 'mac': Tr	{'s
'min	['Valve']	['Valve']		{'currency	[29]	{'windows': True, 'mac': True,	
'min	['Valve']	['Valve']		{'currency	[30]	{'windows': True, 'mac': Tr	{'s
'min	['Valve']	['Valve']		{'currency	[31]	{'windows': True, 'mac': True,	
'min	['Gearbox Software']	['Valve']		{'currency	[32]	{'windows': True, 'mac': True,	
'min	['Valve']	['Valve']		{'currency	[33]	{'windows': True, 'mac': True,	
'min	['Valve']	['Valve']		{'currency	[34, 29	{'windows': True, 'mac': Tr	{'s
]	['Valve']	['Valve']		{'currency	[7]	{'windows': True, 'mac': Tr	{'s
'min	['Gearbox Software']	['Valve']		{'currency	[35]	{'windows': True, 'mac': Tr	{'s
'min	['Valve']	['Valve']	[{'appid': :	{'currency	[36, 28	{'windows': True, 'mac': Tr	{'s
'min	['Valve']	['Valve']		{'currency	[37]	{'windows': True, 'mac': Tr	{'s
'min	['Valve']	['Valve']		{'currency	[38]	{'windows': True, 'mac': True,	
'min	['Valve']	['Valve']		{'currency	[25]	{'windows': True, 'mac': Tr	{'s
'min	['Valve']	['Valve']		{'currency	[39, 79	{'windows': True, 'mac': True,	
]	['Valve']	['Valve']				{'windows': True, 'mac': True,	
'min	['Valve']	['Valve']		{'currency	[38]	{'windows': True, 'mac': True,	
'min	['Valve']	['Valve']		{'currency	[79, 46	{'windows': True, 'mac': Tr	{'s
'min	['Valve']	['Valve']	[{'appid': 4	{'currency	[515, 2	{'windows': True, 'mac': Tr	{'s
'min	['Valve']	['Valve']		{'currency	[516, 4	{'windows': True, 'mac': Tr	{'s
'min	['Valve']	['Valve']			[19784	{'windows': True, 'mac': Tr	{'s
'min	['Valve']	['Valve']		{'currency	[1053,	{'windows': True, 'mac': Tr	{'s
'min	['Valve']	['Valve']		{'currency	[2481,	{'windows': True, 'mac': Tr	{'s
'min	['Valve']	['Valve']			[19784	{'windows': True, 'mac': Tr	{'s
'min	['Valve']	['Valve']		{'currency	[7877,	{'windows': True, 'mac': Tr	{'s
]	['Valve']	['Valve']				{'windows': True, 'mac': Fa	{'s
'min	['Valve', 'Hidden Path E	['Valve']			[32938	{'windows': True, 'mac': Tr	{'s
]	['Mark Healey']	['Mark He	[{'appid': :	{'currency	[45]	{'windows': True, 'mac': Fa	{'s

Merging and cleaning in Kaggle Steam Dataset

- Merged relevant columns of steam_app_data.csv and steamspy_data.csv
- Drop rows which has missing data or duplicates or fill with blank / appropriate data
- Convert dates to date time format
- Fixing headers for columns
- Since some data is in dictionary, we used ast.literal_eval to convert them to strings and joined them together with ";"
- Create rating variable using Wilson Score Interval from yes/no recommendations

THE DATASET

	name	steam_appid	controller_support	dlc	short_description	demos	platforms	movies	achievements	release_date	...	developer	publisher	owners	average_forever	median_forever	initialpri
0	Counter-Strike	10	0	0	Play the world's number 1 online action game. ...	0	windows;mac;linux	0	0	2000-11-01	...	Valve	Valve	10,000,000 .. 20,000,000	17612	317	9.
1	Team Fortress Classic	20	0	0	One of the most popular online action games of...	0	windows;mac;linux	0	0	1999-04-01	...	Valve	Valve	5,000,000 .. 10,000,000	277	62	4.
2	Day of Defeat	30	0	0	Enlist in an intense brand of Axis vs. Allied ...	0	windows;mac;linux	0	0	2003-05-01	...	Valve	Valve	5,000,000 .. 10,000,000	187	34	4.
3	Deathmatch Classic	40	0	0	Enjoy fast-paced multiplayer gaming with Death...	0	windows;mac;linux	0	0	2001-06-01	...	Valve	Valve	5,000,000 .. 10,000,000	258	184	4.
4	Half-Life: Opposing Force	50	0	0	Return to the Black Mesa Research Facility as ...	0	windows;mac;linux	0	0	1999-11-01	...	Gearbox Software	Valve	5,000,000 .. 10,000,000	624	415	4.

CLEANING THE DATASET

Merging of Steam User Data and Steam API calls

- Used google cloud platform to host Steam User Data for SQL. Connected to the database and fetched data with conditions and exported it to CSV

- Called Steam's Developer API calls to get more information on user's games owned and ratings

- Merged Both dataset together using steam user ID and exported it

```
#Database Connection (dont run this again , too costly!!) #Getting Data
db_connection = mysql.connect(user="root", password="jajasauce", host="34.143.214.112", database="steam")
cur = db_connection.cursor()

cur.execute("SELECT steamid,personaname FROM Player_Summaries WHERE personastate != 0 LIMIT 100000;") # We shall only limit it to 10
myresult = cur.fetchall()
df = pd.DataFrame(myresult,columns=['steamid','personaname'])
df.to_csv('playerinfo.csv', index=False) #We exported it to a csv file

API = "4141070D32E9CF793B1D9BC8A25C5950"

steam_data = pd.read_csv("SteamUserData/playerInfo.csv",nrows=20000) #We found out that 100,000 data is too much because each user c

#Getting list of User game library data
data1 = {'steamid':[],'gamesid':[],'playtime_forever':[]}
dfGames = pd.DataFrame(data1)

for x in steam_data['steamid']:
    response = requests.get("http://api.steampowered.com/IPlayerService/GetOwnedGames/v0001/?key="+API+"&steamid="+str(x)+"&format=j
    reply = response.json();
    if(reply['response'] == {}):
        continue
    else:
        for y in reply['response']['games']: # save it to data frame instead
            new_row = {'steamid':str(x),'gamesid':y['appid'],'playtime_forever':y['playtime_forever']}
            dfGames = dfGames.append(new_row, ignore_index=True)


dfGames.to_csv('../SteamUserData/playerGames.csv', index=False)
```

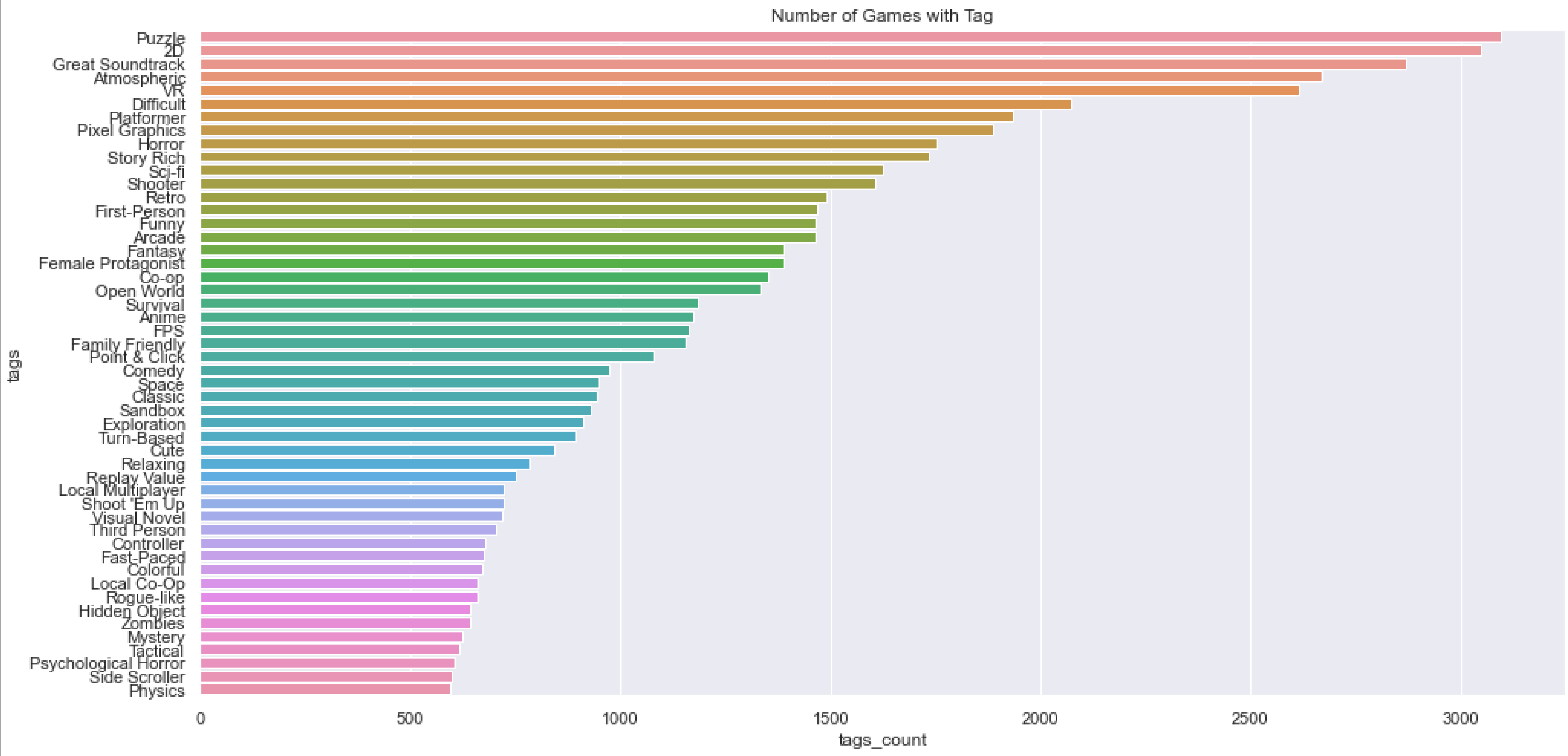
	steamid	gamesid	playtime_forever	appType	price	rating	is_Multiplayer	FriendHasGame
0	76561197960269742	10.0	0.0	game	9.99	4	1	1
1	76561197960270817	10.0	0.0	game	9.99	1	1	1
2	76561197960270881	10.0	101.0	game	9.99	5	1	1
3	76561197960271173	10.0	1442.0	game	9.99	3	1	1
4	76561197960271217	10.0	101.0	game	9.99	5	1	1
...	...	...	...	...	...	...	...	...
171236	76561197960354066	11900.0	51.0	game	9.99	5	0	1
171237	76561197960354971	11900.0	297.0	game	9.99	5	0	1
171238	76561197960355570	11900.0	1.0	game	9.99	2	0	1
171239	76561197960359884	11900.0	0.0	game	9.99	3	0	1
171240	76561197960369216	11900.0	30.0	game	9.99	4	0	1

171241 rows × 8 columns

IMPORTANT VARIABLES IN THE DATASET

Gaming Keywords	Game Rating	Game Quality	Game Genres	Correlation Matrix

EXPLORATORY DATA ANALYSIS

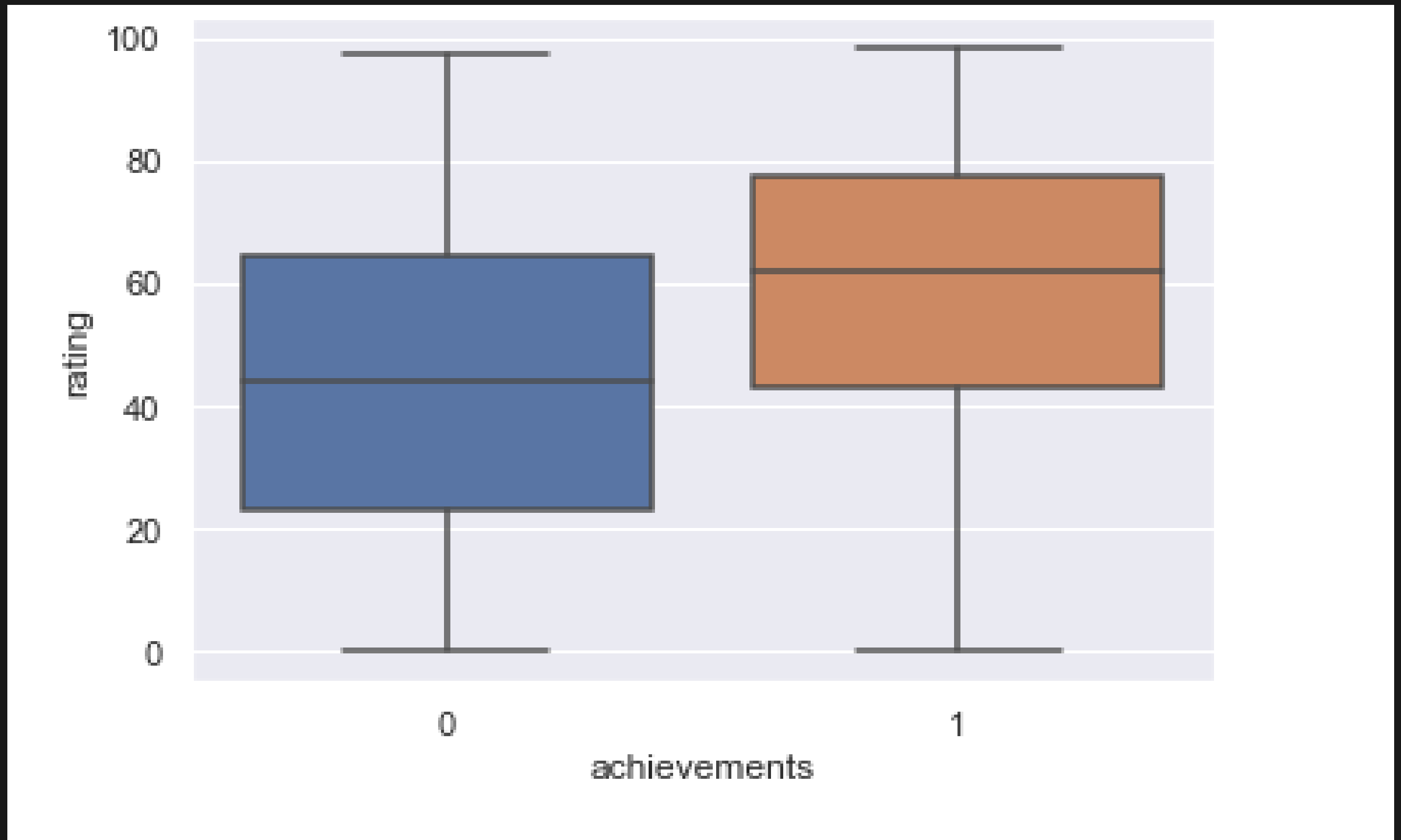


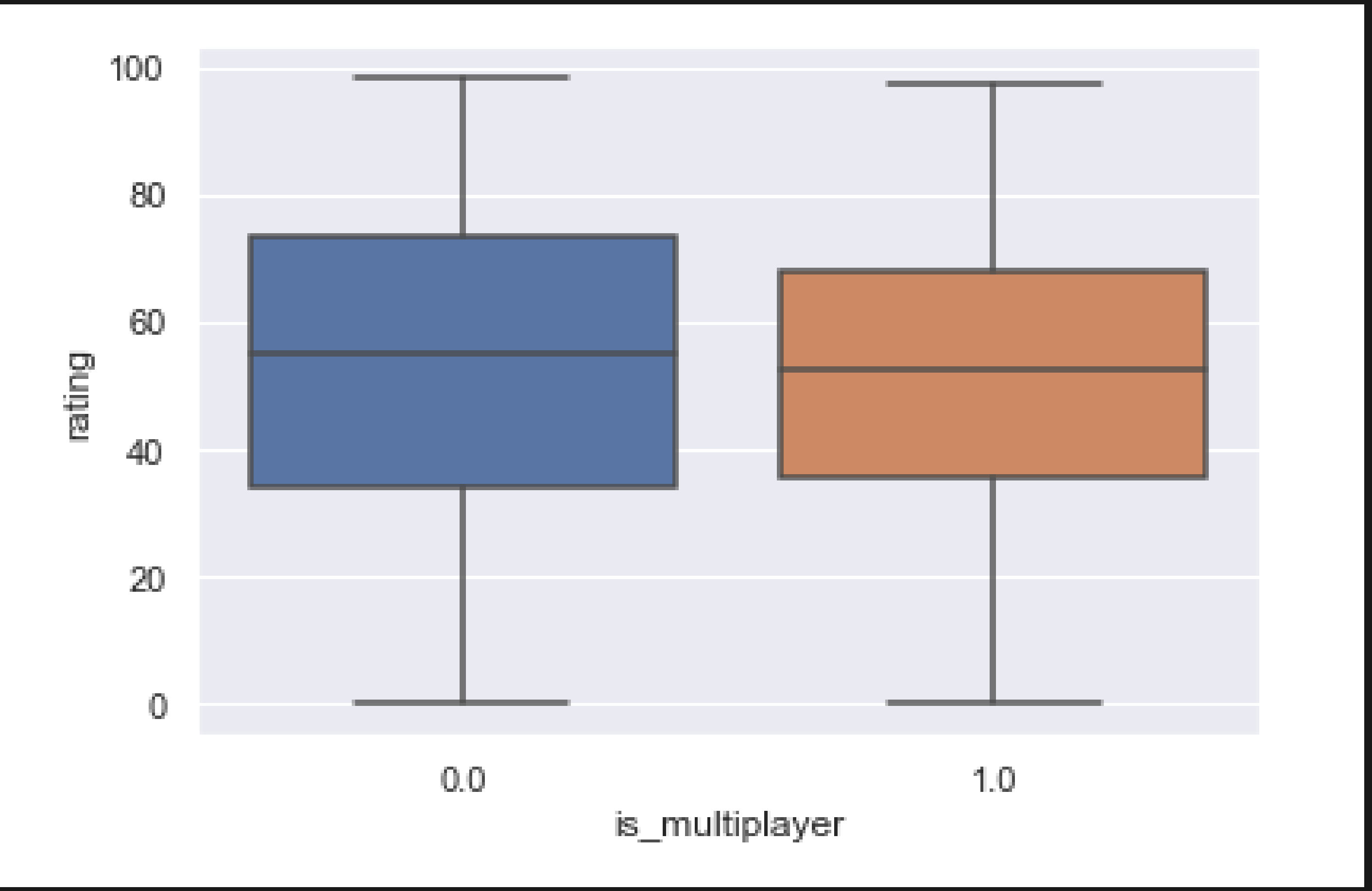
Gaming Keywords

- Made a word cloud of game descriptions to grasp a better understanding of words used in the description

- Used a bar plot to see the frequency of tags on all the games
- Most of the games are puzzle, 2D games

EXPLORATORY DATA ANALYSIS



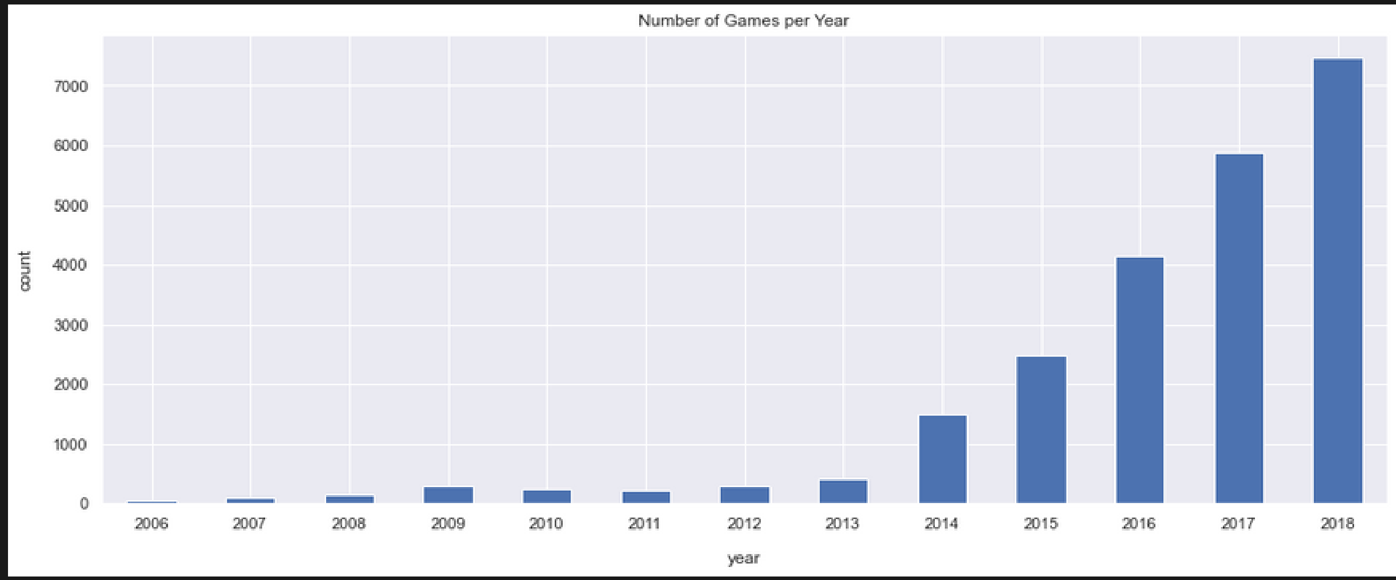


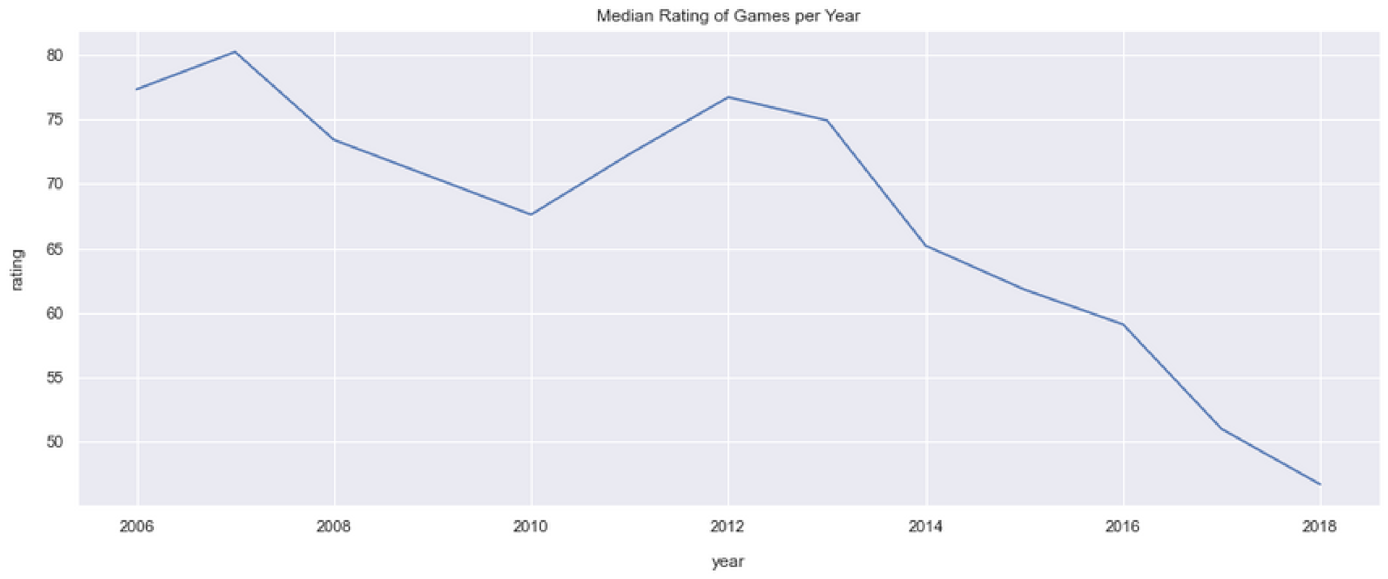
Game Rating

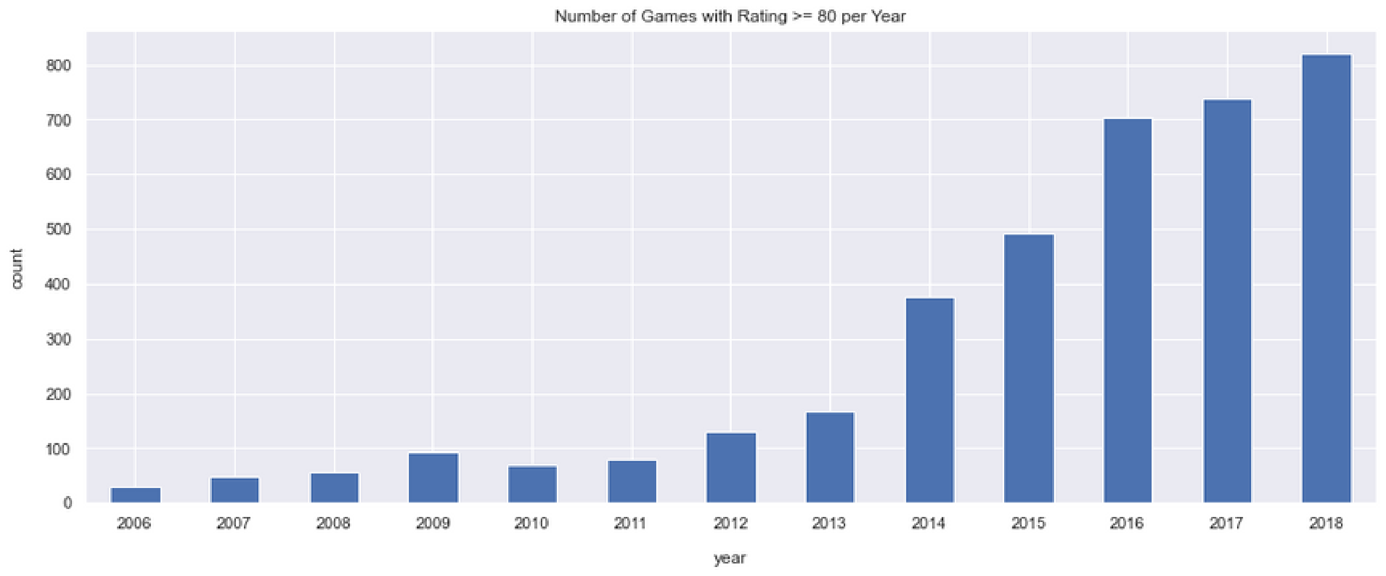
- Used boxplots to visualise the relationship between achievement and is_mulitplayer against ratings
- When a game has achievements, on average, there is a higher chance of achieving a higher rating
- Might be able to use achievements as a variable for our models

- With multiplayer, we see that there is not much impact on ratings

EXPLORATORY DATA ANALYSIS







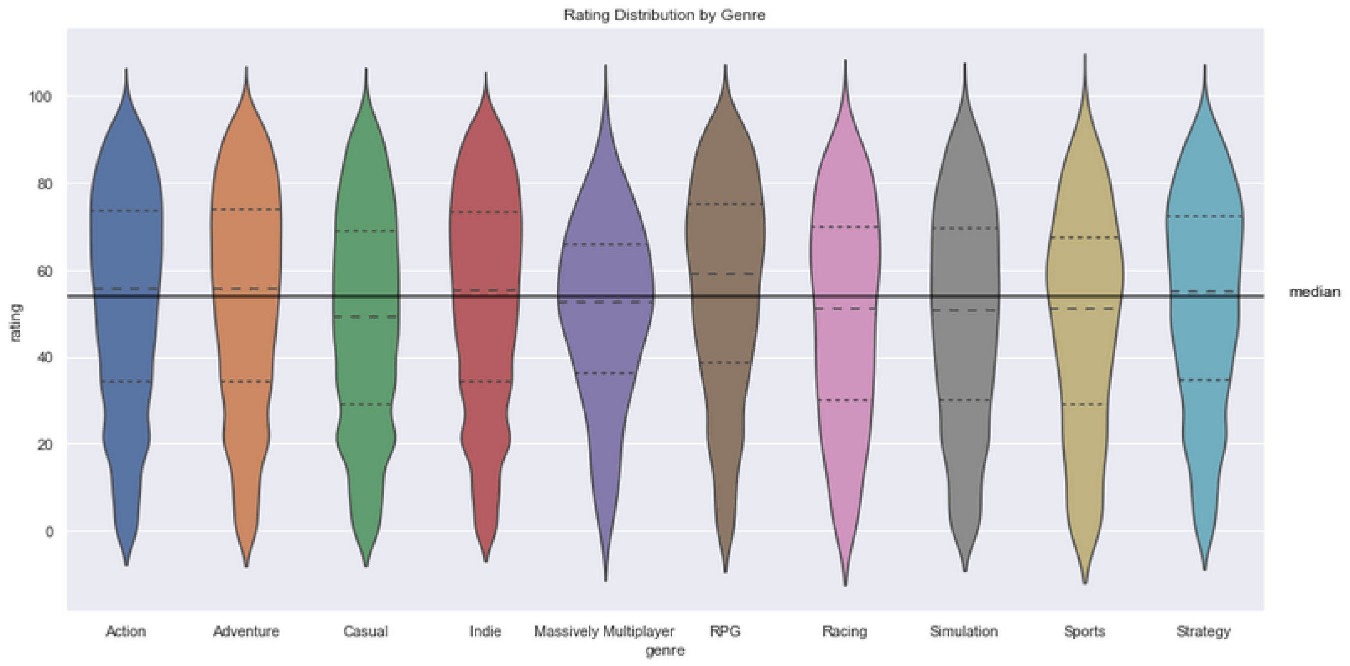
Game Quality

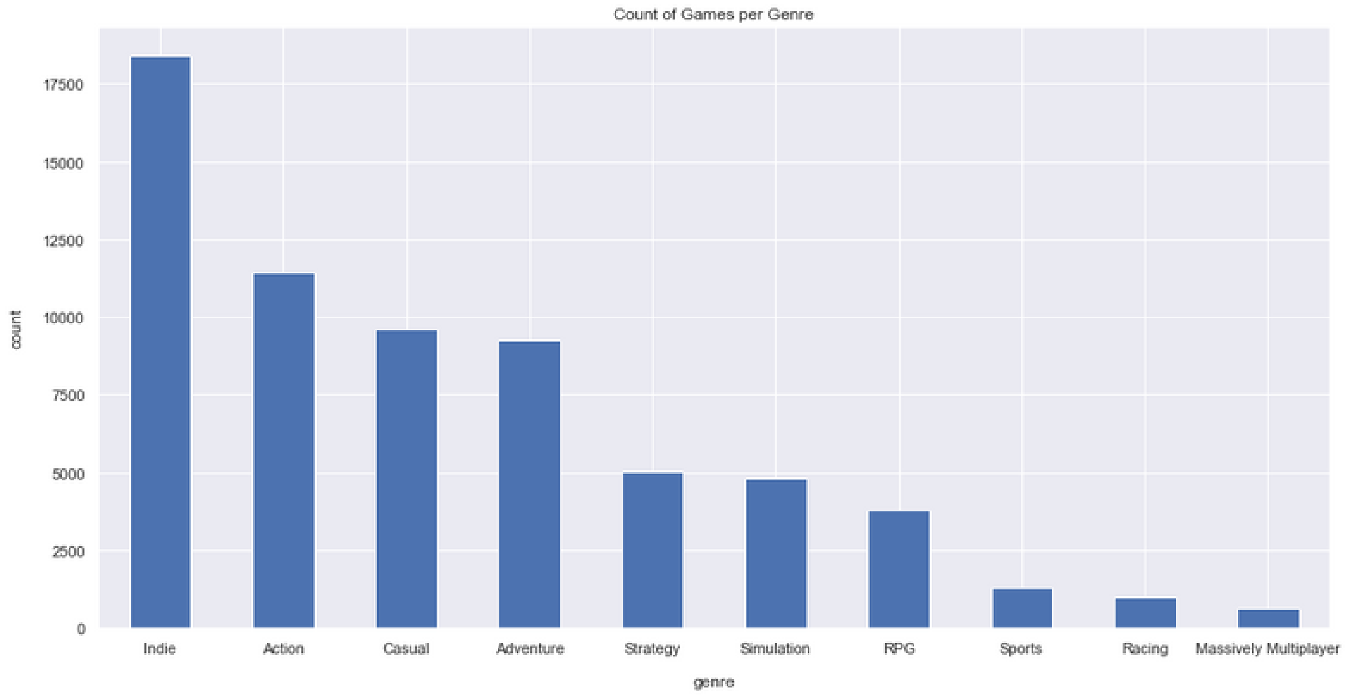
- Visualising number of games released per year in a bar graph
- Clear sign of booming game industry

- Visualising median rating of games per year in a line graph
- Shows declining median quality of games

- Visualising games with rating above 80 per year in a bar graph
- Shows a rise in quality games over the years

EXPLORATORY DATA ANALYSIS



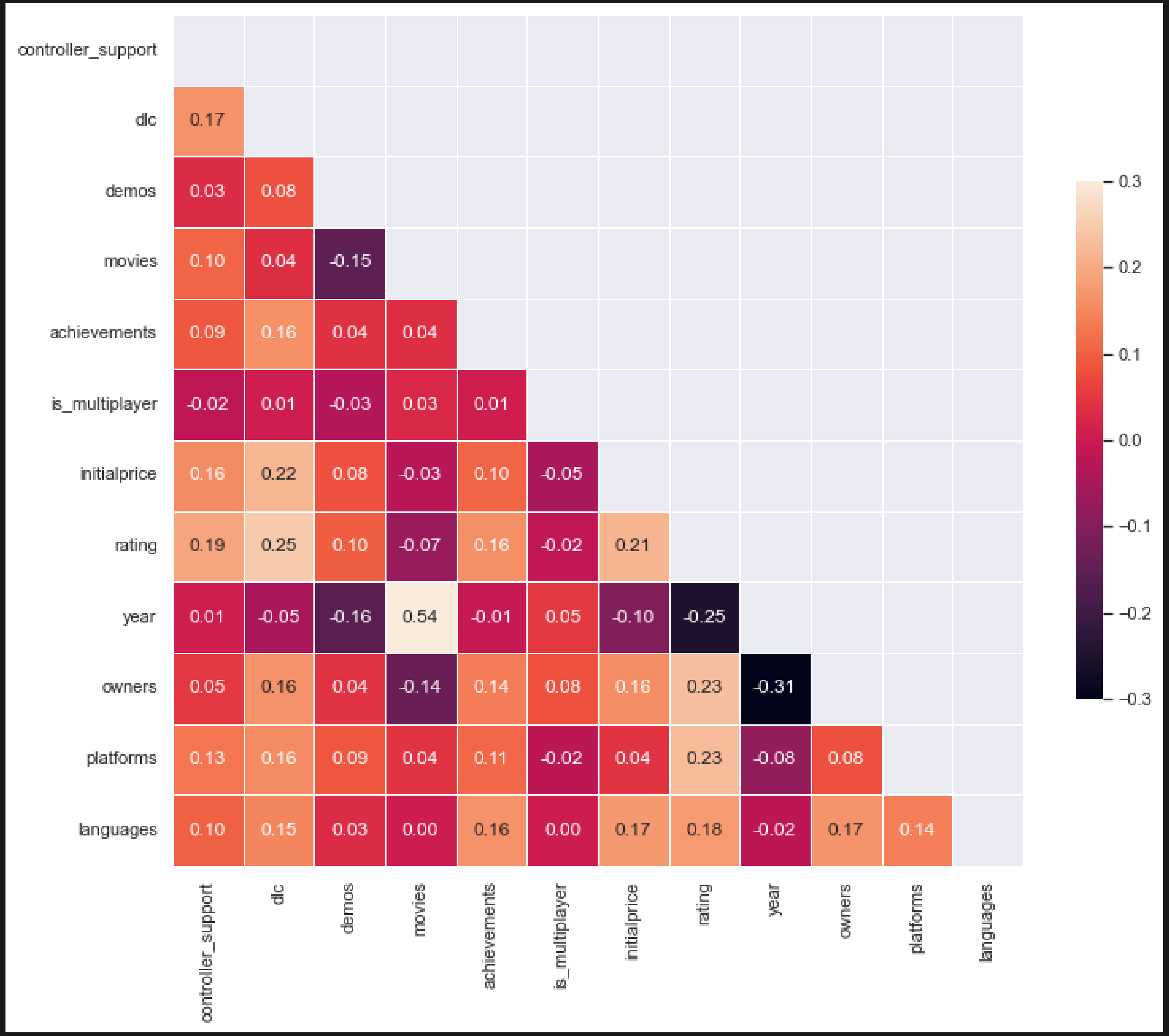


Game Genres

- Visualising genre distribution against rating with a violin plot
- Shows that there is little to no correlation between genres and rating

- Visualising count of games per genre in a bar chart
- Highest counts: Indie, Casual, Adventure

EXPLORATORY DATA ANALYSIS



Correlation Matrix

- Correlation matrix of the variables in our dataset except for genre
- Notable variables with correlation are controller_support, dlc, demos, achievements, initialprice, platforms, languages
- Surprisingly, whether a game is_multiplayer has almost no correlation to its rating.

SOLUTIONS

Rating Predictor

Game Recommendation System

RATING PREDICTOR

Regression Models

- Linear Regression
- KNN Regression
- Random Forest Regression
- Gradient Boosting + Optimizing Hyperparameters
- Including TF-IDF

Classification Models

- Logistic Regression
- Random Forest Classification

PREPARING DATASET

	score
Great Soundtrack	2214.398376
dlc	1644.280463
platforms	1348.745638
initialprice	1205.006330
2D	1110.721748
Story Rich	1049.246363
controller_support	967.448950
Pixel Graphics	850.934465
Atmospheric	845.302054
languages	811.481991
achievements	665.277301
Funny	658.163198
Puzzle	639.309370
Female Protagonist	621.581829
Difficult	603.227652
Classic	598.224882
Co-op	595.523588
Comedy	531.857543
Anime	493.636141
Sci-fi	454.539246

Feature selection, K-fold cross validation

- Use Feature Selection to reduce the number of our input variables using SelectKBest
- Reduce the computational cost of modeling
- Standardize the data using StandardScaler to account for input values with differing scales

- 80:20 train_test_split for our data
- K-fold Cross Validation partitions the data to build a more generalized model

REGRESSION MODEL

```python
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression

scaler = StandardScaler()

X_train = pd.DataFrame(scaler.fit_transform(X_train), columns=X.columns)
X_test = pd.DataFrame(scaler.transform(X_test), columns=X.columns)

lr = LinearRegression()
lr.fit(X_train, y_train)
```

Linear regression

- Linear model that assumes a linear relationship between the input variables X and the output variable y

- LR R^2 (train): 0.241
- LR R^2 (test): 0.224
- LR RMSE (test): 22.373

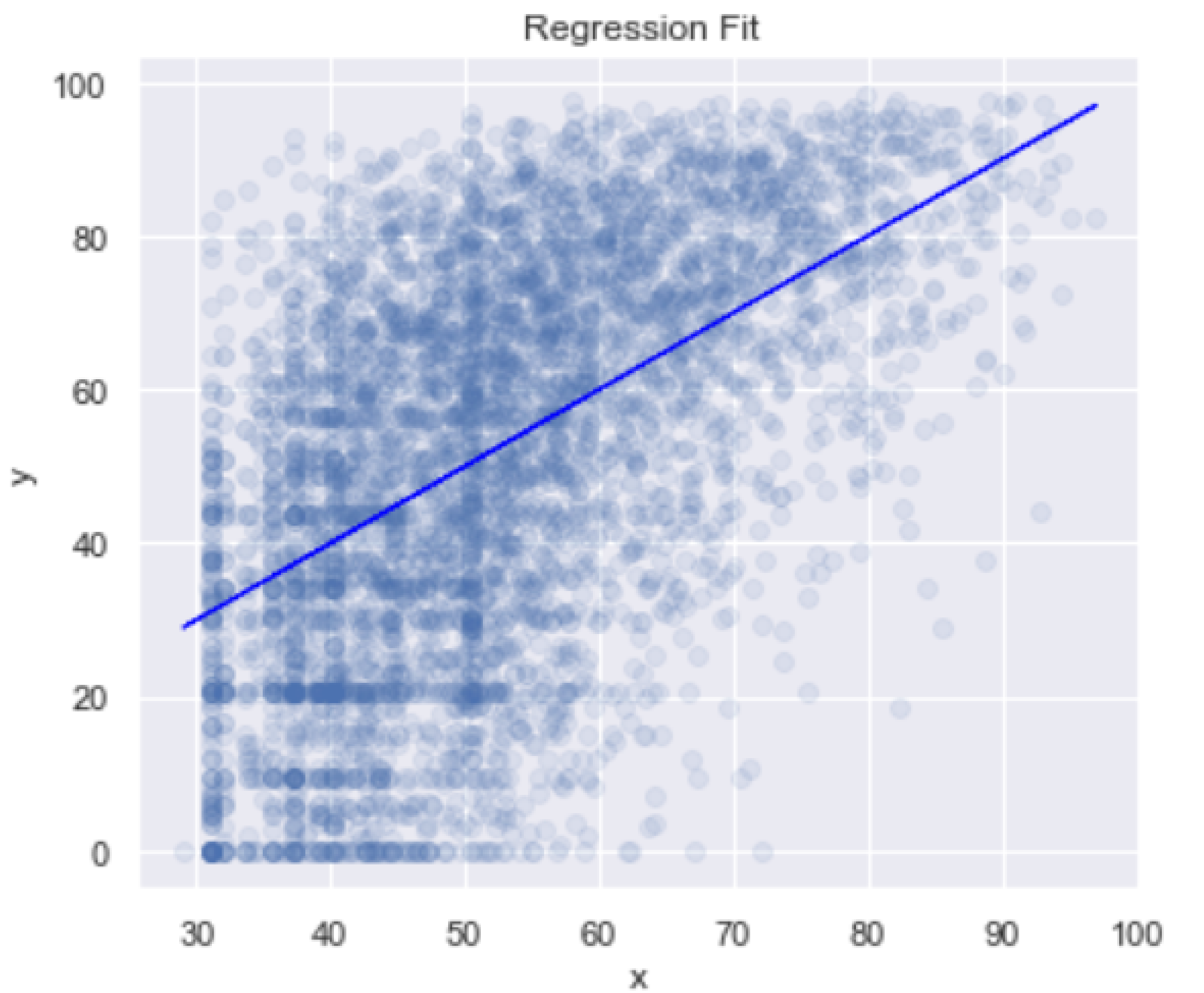
- Low accuracy on train & test indicates underfitting of data

- Chosen regression models may not be suitable to capture the relationship
- Points in our dataset have too much variation.
- Not much strong correlation between rating and the predictors chosen

REGRESSION MODEL

```
from sklearn.ensemble import GradientBoostingRegressor

gbr = GradientBoostingRegressor(random_state = 69)
gbr.fit(X_train, y_train)
```



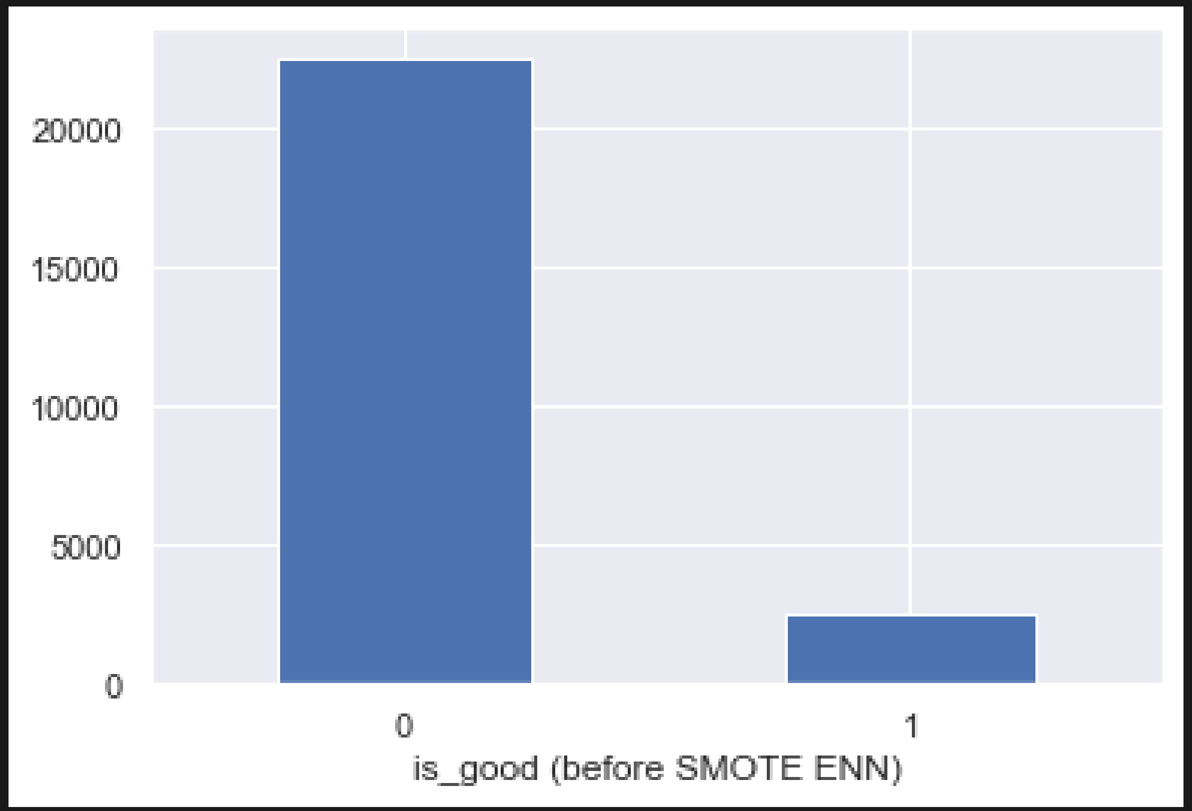
Gradient boosting

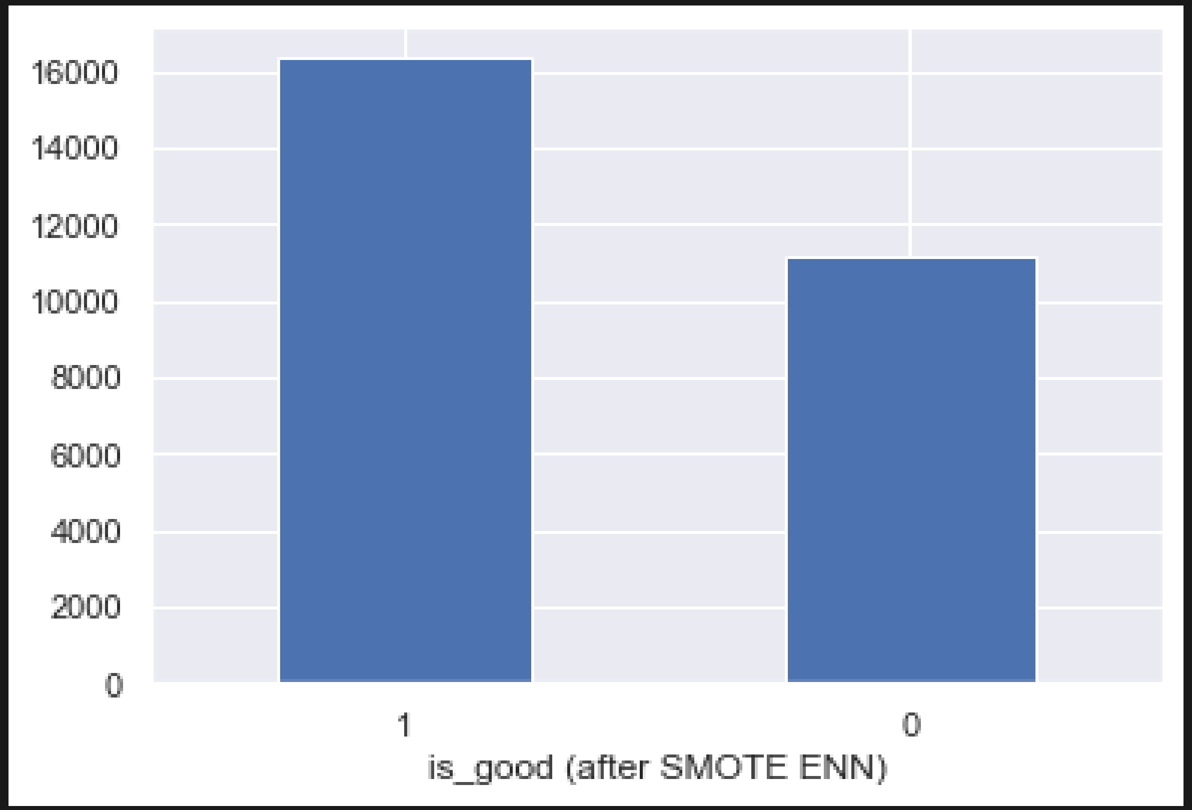
- Uses a loss function to be optimized, a weak learner (eg. decision trees) to make predictions, and an additive model (gradient descent) to add weak learners to minimize the loss function

- GBR R^2 (train): 0.304
- GBR R^2 (test): 0.288
- GBR RMSE (test): 21.334

- Higher R^2 and lower RMSE values compared to Linear Regression
- Better model for our dataset as compared to Linear Regression
- Optimised hyperparameters using GridSearchCV

CLASSIFICATION MODEL



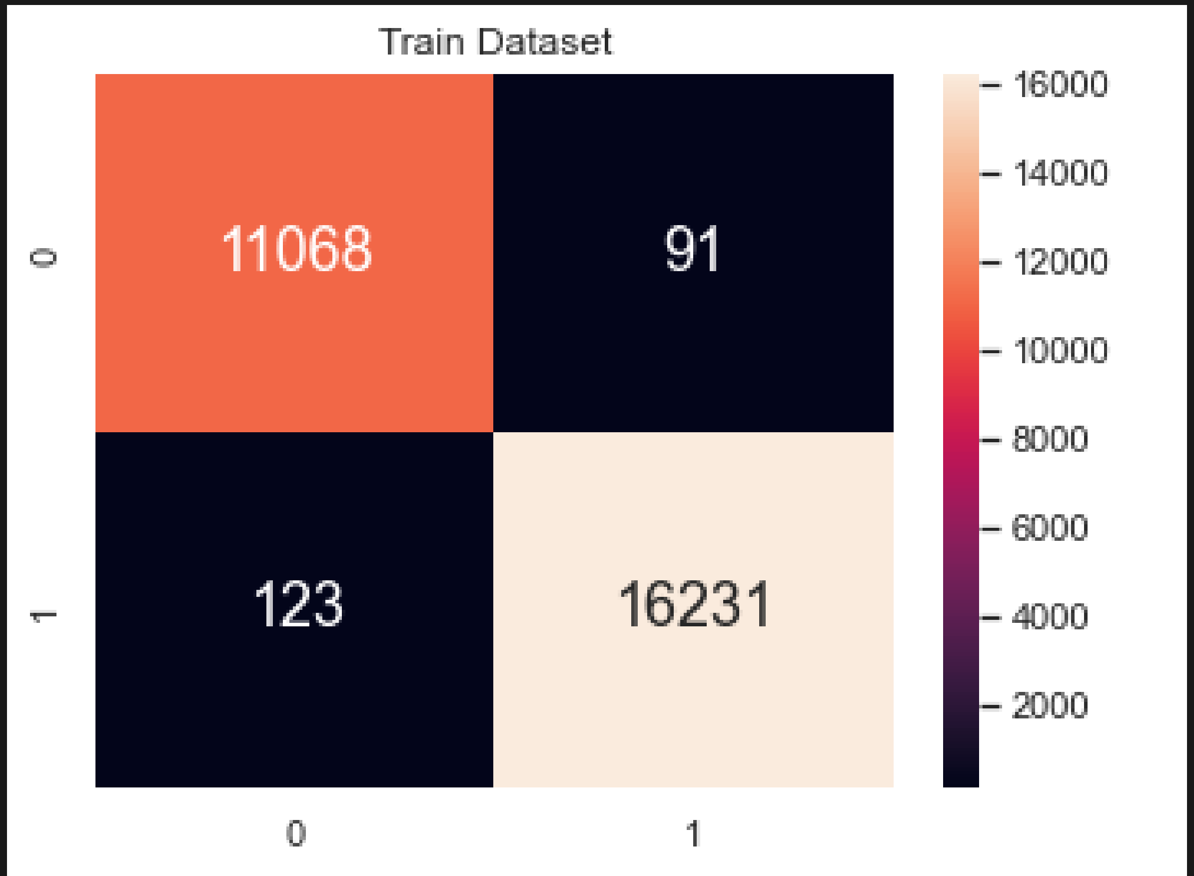


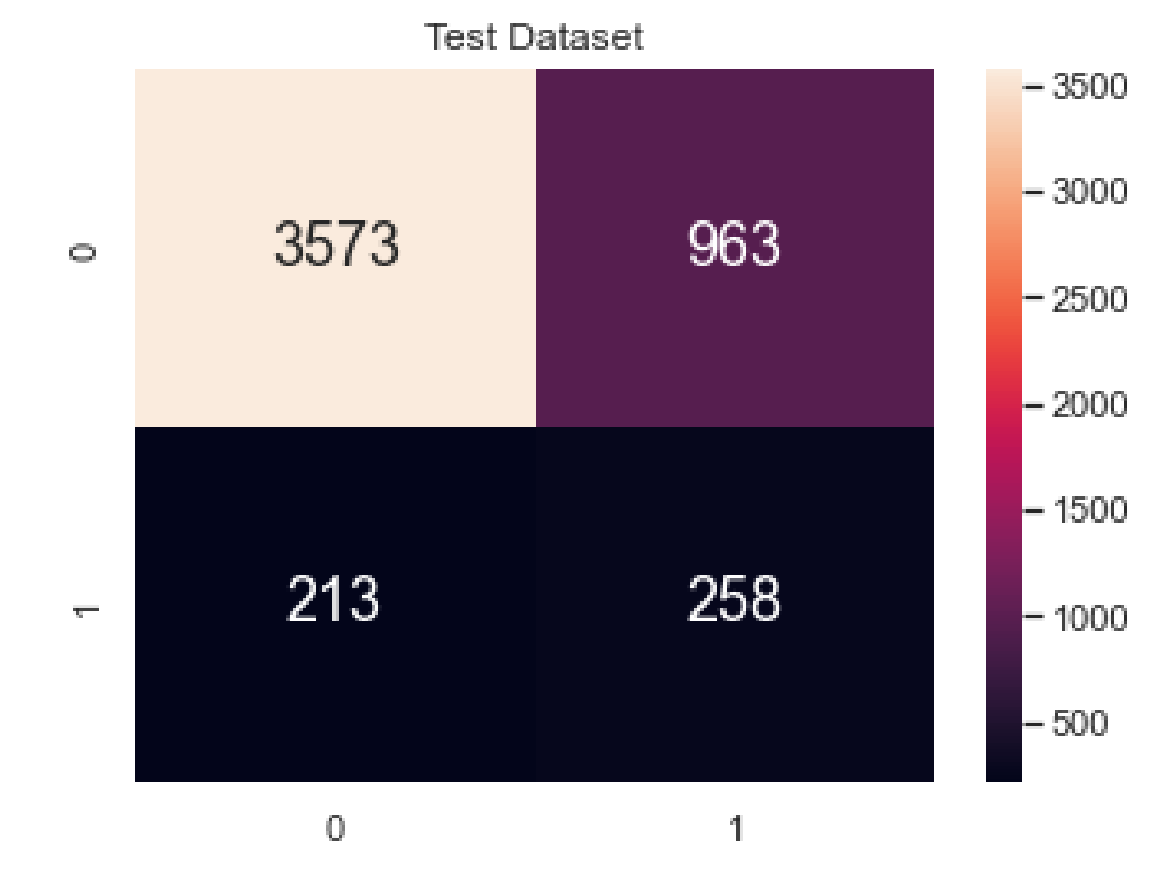
Resampling data using SMOTE ENN

- Create new column in dataset for classification - 'is_good' based on ratings
- However, unequal distribution of classes will reduce performance of models

- Resampling is used to mitigate this issues
- Synthetic Minority Oversampling Technique (oversampling) + Edited Nearest Neighbor (undersampling)

CLASSIFICATION MODEL





Random forest classifier

- Fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting

- RFC Accuracy (train): 0.99
- RFC f1-score (train): 0.99
- RFC Accuracy (test): 0.77
- RFC f1-score (test): 0.3

- Perform better than Logistic Regression for classification model
- Higher accuracy than regression models

GAME RECOMMENDATION SYSTEM

Content-based recommendation

- Used gaming metadata such as game plot / description, developers, related genres, platforms
- Give a score based on similarity

Collaborative filtering recommendation

- Used game rating of all users
- Estimation of all user's gaming "taste"

GAME RECOMMENDATION SYTEM

short_description	genres	
Play the world's number 1 online action game. ...	Action	Old School;Surv
)ne of the most popular online action games of...	Action	Old School;Fast-Pac
Enlist in an intense brand of Axis vs. Allied ...	Action	Historical;Class-E
:njoy fast-paced multiplayer gaming with Death...	Action	First-Person;Class
Return to the Black Mesa Research Facility as ...	Action	Silent Protagonist;I
...	...	
The Room of Pandora is a third-person interact...	Adventure;Casual;Indie	
Cyber Gun is a hardcore first-person shooter w...	Action;Adventure;Indie	Cyberpunk;Fast-Pa
Super Star Blast is a space based game with ch...	Action;Casual;Indie	

$$Cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \, \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2} \sqrt{\sum_1^n b_i^2}}$$

```
Enter name of your favourite game: Counter-Strike
Since you liked Counter-Strike, you should also try:
                                 name  similarity
1                Team Fortress Classic    0.503643
2                 Day of Defeat: Source    0.498058
3    Counter-Strike: Global Offensive    0.473155
4               Counter-Strike: Source    0.462329
5                           Insurgency    0.436205
6                           Death Toll    0.421159
7                          Alien Swarm    0.420237
8      Counter-Strike: Condition Zero    0.413665
9                      Team Fortress 2    0.410877
10                             Undoing    0.407120
```

Content-based Recommendation

- Use the steam dataset to make recommendations based on contents
- Use text data containing short description, genres, additional_tags, developer, publisher, platforms of games
- Further cleaning of data by nltk library, removing spaces, joining variables and removing stop words such as "like" "a" "the".
- Using CountVectorizer (Sklearn) to vectorized text data, and calculate the Cosine Similarity of that particular game with all games.
- The top most similar games will then be recommended

GAME RECOMMENDATION SYTEM

	steamid	gamesid	playtime_forever	appType	price	rating	is_Multiplayer	FriendHasGame
0	76561197960269742	10.0	0.0	game	9.99	4	1	1
1	76561197960270817	10.0	0.0	game	9.99	1	1	1
2	76561197960270881	10.0	101.0	game	9.99	5	1	1
3	76561197960271173	10.0	1442.0	game	9.99	3	1	1
4	76561197960271217	10.0	101.0	game	9.99	5	1	1
...	...	...	...	...	...	...	...	...
171236	76561197960354066	11900.0	51.0	game	9.99	5	0	1
171237	76561197960354971	11900.0	297.0	game	9.99	5	0	1
171238	76561197960355570	11900.0	1.0	game	9.99	2	0	1
171239	76561197960359884	11900.0	0.0	game	9.99	3	0	1
171240	76561197960369216	11900.0	30.0	game	9.99	4	0	1

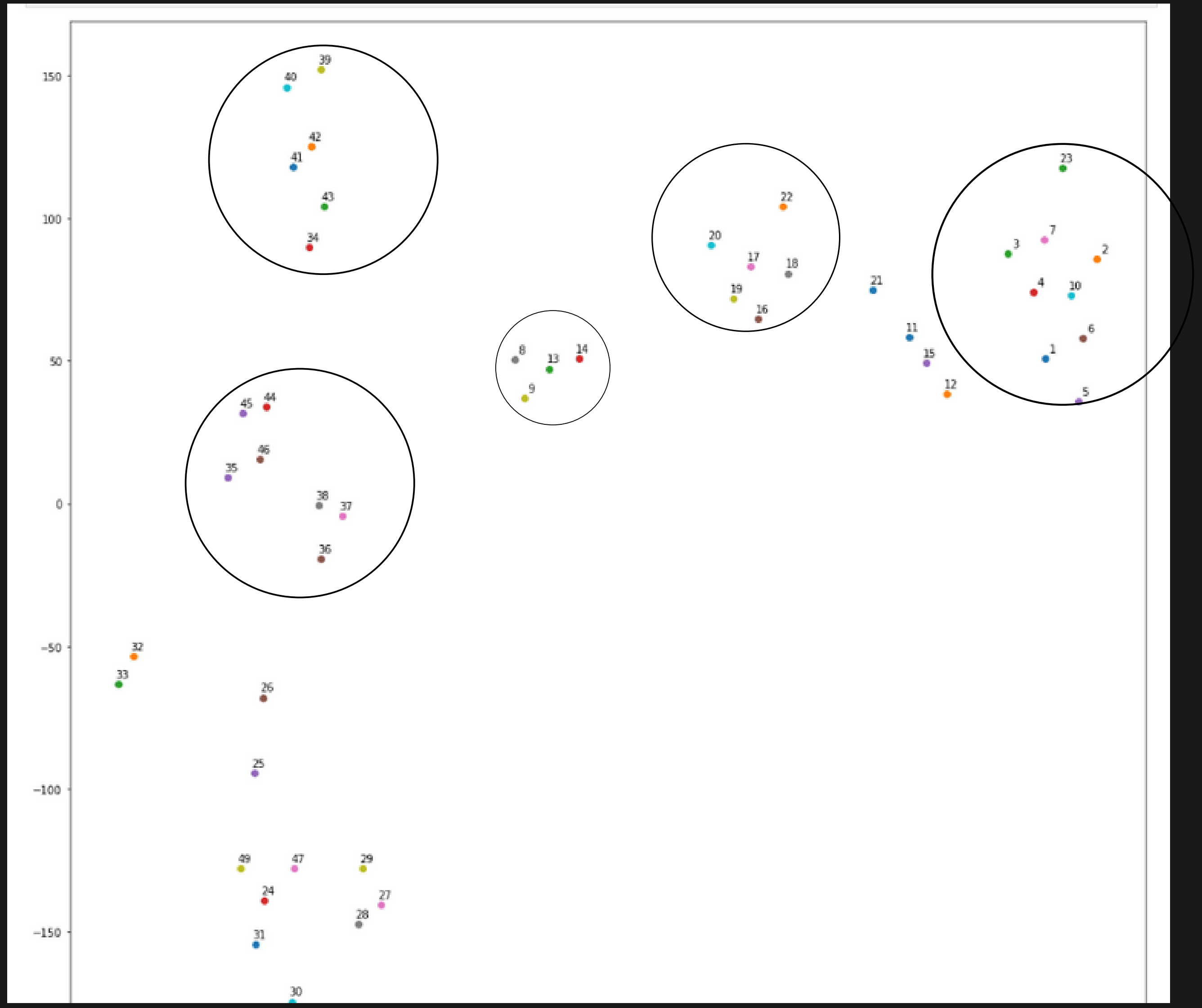
171241 rows × 8 columns

gamesid	10	20	30	40	50	60	70	80	100	130	...	433850	434570	439190	443080	446620	448280	450540	45152
steamid																			
76561197960269742	4	2	3	2	5	3	2	4	2	5	...	0	0	0	0	1	4	3	3
76561197960270817	1	1	2	1	3	4	1	4	3	3	...	0	0	2	0	0	0	3	0
76561197960270881	5	4	5	2	2	2	4	3	5	4	...	3	0	0	4	0	0	0	0
76561197960271173	3	3	3	5	3	4	4	0	0	3	...	2	4	0	0	0	0	0	0
76561197960271217	5	4	1	4	2	5	2	2	4	3	...	4	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
76561197960410700	5	1	4	1	3	4	4	0	0	3	...	0	0	0	0	0	0	0	0
76561197960412986	3	3	3	4	3	4	1	0	0	4	...	0	0	5	0	0	0	0	0
76561197960413532	4	4	4	3	4	4	5	3	3	1	...	0	0	0	0	0	0	0	0
76561197960417000	5	3	5	3	4	3	2	0	0	5	...	0	0	2	0	0	0	0	0
76561197960418886	5	5	2	3	5	4	3	4	3	3	...	0	0	0	0	0	0	0	0

Collaborative Filtering Recommendation

- Steam User Data which was called using our SQL server and steam's API calls
- Used Truncated Singular Value Decomposition on our data (UserID, rating, gamesid)
- Builds a model based on the past behaviour of users. In this way, the model finds an association between the users and the items.
- Model is then used to predict the rating for the games in which the user may be interested

COLLABORATIVE FILTERING RECOMMENDATION



Singular Value Decomposition (SVD) (Truncated)

- SVD decomposes a matrix into constituent arrays of feature vectors corresponding to each row and each column
- Able to better estimate the ratings of user and the matrix will then represent a generalized view of users' "tastes"
- Visualising our data using t-Distributed Stochastic Neighbor Embedding (t-SNE), we can see that SVD is finding points close to each other within different dimensions and grouping them up

COLLABORATIVE FILTERING RECOMMENDATION

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

```
Since you liked ['Portal 2', 'Half-Life', 'Counter-Strike'], you should also try:
                     name  correlation
0                  Portal     0.756516
1              Half-Life 2    0.725950
2           Left 4 Dead 2     0.725193
3       Deathmatch Classic    0.881013
4     Team Fortress Classic   0.875369
5                Ricochet     0.872527
6  Half-Life: Opposing Force  0.874300
```

Pearson's correlation coefficient

- Calculate their similarities by using Pearson's correlation coefficient for each game
- Recommend games with the highest correlation to the user's "taste "

CONCLUSION

DATA-DRIVEN INSIGHTS

Classification VS Regression

- Easier to predict discrete values rather than continuous values
- Unable to accurately predict for Regression

Best Models

- Gradient Boosted Regression for Regression
- Random Forest Classifier for Classification

Insufficient Data

- Missing factors such as budget of the game
- Exponential increase in games over the years create possibility of skewed data

LEARNING OUTCOME

SQL server and API usage

- Set up Google cloud SQL server
- SQL queries
- Calling API of steam's development data

Methods to handle data

- SMOTE
- SelectKBest
- StandardScaler
- Statistical methods (Wilson Score, Bayesian Averaging)

New models

- KNN
- Logistic Regression
- Random Forest
- Gradient Boosting
- Truncated SVD, t-SNE
- TF-IDF, Count Vectorizer

FINAL OUTCOME

Prediction for Rating

- Even with classification not ideal accuracy
- Shows that Ratings are volatile to external factors
- Can be used as a gauge for both gamers and game creators

Recommendation

- Recommendation system able to show users which game is suited for their tastes
- Gamers can now filter out games for themselves

ARE YOU READY
TO FIND THE PERFECT GAME?

Detailed walkthrough notebooks at
https://github.com/bryan9898/1015