Machine Learning and Predicting Video Game Hits

Andrew Marino, Solongo Tserenkhand, Robin Bun, Sarah Zachrich, Kevin Schram, Madison Chamberlain

Our Questions:

- What features or traits of a videogame determine whether it will be a “hit”?
- How accurate can a game’s popularity be predicted?

Our Audience:

- Game Developers and Publishers

Our Data Sources:

- Steam: Video Game and Software Marketplace
- Steam API
- Web Scrapes of SteamSpy, and SteamDB

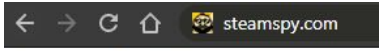
What is Steam?

Steam is a digital distribution service for PC games. This cloud-based gaming library allows user to access purchased games from a content delivery network, and also match with other users for multiplayer gaming. Steam has over 150 million users across 194 countries.

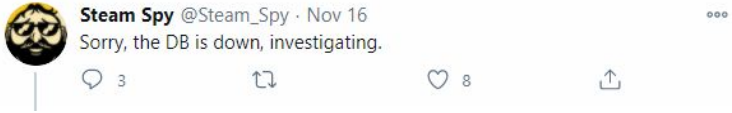
Web Scraping

```
try:
    tag2 = []
    tags = tags.findNext()
    while (more_tags == 1):
        #print(tags.get_text())
        if (tags.get_text() == ""):
            more_tags = 0
        else:
            tag2.append(tags.get_text())
            tags = tags.findNext()
    list_of_tags.append(tag2)
```

```
3]: #steam spy blocks automated web scraping so we need to pass it a header telling it that we are a human using a regular web bro
header = {
  "User-Agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.75 Safari/537.36",
  "X-Requested-With": "XMLHttpRequest"
}
```



Connection failed: Too many connections



	A	B	
1	App Id	Name	Dev
2	7	Steam Client	
3	8	winui2	
4	10	Counter-Strike	Val
5	20	Team Fortress Classic	Val
6	30	Day of Defeat	Val
7	40	Deathmatch Classic	Val
8	50	Half-Life: Opposing Force	Gea
9	60	Ricochet	Val
10	70	Half-Life	Val
95240	1474610	The Falconeer - Game Manual	Ton
95241	1474820	The Tower Of TigerQiuQiu Tiger Tan	Tige
95242	1474960	Der Geisterturm / The Ghost Tower	Gra
95243	1475910	DIY Simulator Dedicated Server	
95244	1476030	OTTI'S QUEST	
95245	2028850	Bioshock Infinite: Columbia's Finest	Irra

Cleaning our Data

[3]:

	App Id	Name	Developers	Publishers	Metascores	Owners	Genres	Tags	Price
0	7	Steam Client	NaN	NaN	NaN	0Â ..Â 20,000	[']	NaN	$0.00
1	8	winui2	NaN	NaN	NaN	0Â ..Â 20,000	[']	NaN	$0.00
2	10	Counter-Strike	Valve	Valve	88%	10,000,000Â ..Â 20,000,000	['Action']	['Action', 'FPS', 'Multiplayer', 'Shooter', 'C...	$9.99
3	20	Team Fortress Classic	Valve	Valve	NaN	2,000,000Â ..Â 5,000,000	['Action']	['Action', 'FPS', 'Multiplayer', 'Classic', 'H...	$4.99
4	30	Day of Defeat	Valve	Valve	79%	5,000,000Â ..Â 10,000,000	['Action']	['FPS', 'World War II', 'Multiplayer', 'Shoote...	$4.99

[4]:
```
#steam_data["Genres"] =
steam_data['clean_genres']=steam_data["Genres"].str.replace(' ','_').str.replace(',_', ',')
```

```
steam_data_clean = steam_data.dropna(subset=["Genres", "Developers","Tags","Metascores", "max owners", "min owners","clean_genres
```

```
steam_data_clean.shape
```

```
(4005, 12)
```

```
steam_data_clean.isna().sum()
```

```
App Id           0
Name             0
Developers       0
Publishers      61
Metascores       0
Genres           0
Tags             0
Price            0
clean_genres     0
clean_tags       0
min owners       0
max owners       0
dtype: int64
```

```
steam_data_clean["min owners"] = pd.Series(steam_data_clean["min owners"]).str.replace(',', "")
steam_data_clean.head()
steam_data_clean["max owners"] = pd.Series(steam_data_clean["max owners"]).str.replace(',', "")
steam_data_clean
```

In [13]:
```
new = steam_data["Owners"].str.split("Â ..Â ", n = 1, expand = True)
steam_data['Owners'].dtypes
new
```

Out[13]:

	0	1
0	0	20,000
1	0	20,000
2	10,000,000	20,000,000
3	2,000,000	5,000,000
4	5,000,000	10,000,000
...	...	...
95239	0	20,000
95240	0	20,000
95241	0	20,000
95242	0	20,000
95243	0	20,000

95244 rows × 2 columns

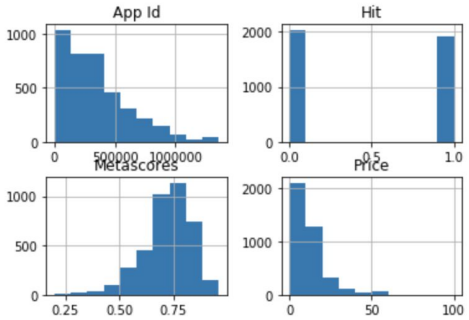
```
steam_data_clean['max owners'] = steam_data_clean['max owners'].astype(int)
```

Our Machine Learning Algorithm

Model used Logistic Regression with Scaled

1. Predict Hit using using price and metorscope
2. Accuracy 0.62
3. Price and Metascores.
4. Load the model

```
dtype=object)
```




```
coe_df = coe_df.sort_values(by=['abs'],
coe_df
```

	coef	feature	abs
2605	3.151072	Price	3.151072
2604	2.946084	Metascores	2.946084

```
# read and binary file
import pickle
import pandas as pd
loaded_model = pickle.load(open("lr_model.pickle", 'rb'))

def predict_hit(real_price, metascore):
    """
        metascore does not need to be scaled
        price needs to be scaled using 69.03, which was the max in the training set
    """
    scaled_price = real_price / 69.03

    # convert data to df
    input_df =  pd.DataFrame(data = [[scaled_price, metascore]] , columns=['Price', 'Metascores'])

    # predict the value
    predictions = loaded_model.predict(input_df)

    # just keep the first item because out df just had one row
    predicted_value = predictions[0]

    return predicted_value
```

Utilizing Machine Learning

1. It is Javascript making calls to Flask

```javascript
<script>
    function predict_hit() {
      price = document.getElementById("priceRange").value;
      metascore = document.getElementById("metascoreRange").value;

      // this is temporary while api is queried
      document.getElementById("isHit").innerHTML = "checking...";

      fetch("/predict/" +  price + "/" + metascore)
        .then(response => response.json())
        .then(result => {
          console.log(result);
          is_hit = result.is_hit;
          if (is_hit == 1) {
            document.getElementById("isHit").innerHTML = "Yes"
          } else {
            document.getElementById("isHit").innerHTML = "No"
          }

          console.log(result.is_hit);
        });

    }
```

```python
# Create route
@app.route('/')
def index():
    """
        Create route that renders machine_learning.html
    """
    return render_template("machine_learning.html")


# Route that trigger display function.
@app.route('/predict/<price>/<metascore>')
def display(price, metascore):
    """

    """
    price = float(price)
    metascore = float(metascore)
    # user sends metascore from 1-100
    # but we need 0.0 - 1.0, so divide by 100
    metascore = metascore / 100.0

    is_hit = predict_hit(real_price=price, metascore=metascore)
    return jsonify({"is_hit": is_hit,
            "price": price,
            "metascore": metascore})

if __name__ == '__main__':
```

Machine Learning
Will my game be a hit?

our data and looking at genres and tags we are able to create a basic filter function that can recommend games based on what you make like. Make your selections below and let the fun begin!
To create this machine learning model we...

Is this a Hit? No
Drag the slider to display the current value.

If you give this model a price and a metascore, it will predict if the game will be a hit (successful) or not.

Price: $50

Metascore: 0.21

```
▶ {is_hit: 0, metascore: 0.16, price: 50}        (index):135
0                                                (index):143
▶ {is_hit: 0, metascore: 0.17, price: 50}        (index):135
0                                                (index):143
▶ {is_hit: 0, metascore: 0.19, price: 50}        (index):135
0                                                (index):143
▶ {is_hit: 0, metascore: 0.21, price: 50}        (index):135
0                                                (index):143
▶ {is_hit: 0, metascore: 0.23, price: 50}        (index):135
0                                                (index):143
▶ {is_hit: 0, metascore: 0.24, price: 50}        (index):135
0                                                (index):143
▶ {is_hit: 0, metascore: 0.25, price: 50}        (index):135
0                                                (index):143
▶ {is_hit: 0, metascore: 0.24, price: 50}        (index):135
0                                                (index):143
▶ {is_hit: 0, metascore: 0.23, price: 50}        (index):135
0                                                (index):143
▶ {is_hit: 0, metascore: 0.22, price: 50}        (index):135
0                                                (index):143
▶ {is_hit: 0, metascore: 0.21, price: 50}        (index):135
0                                                (index):143
>
```

Recommendation App

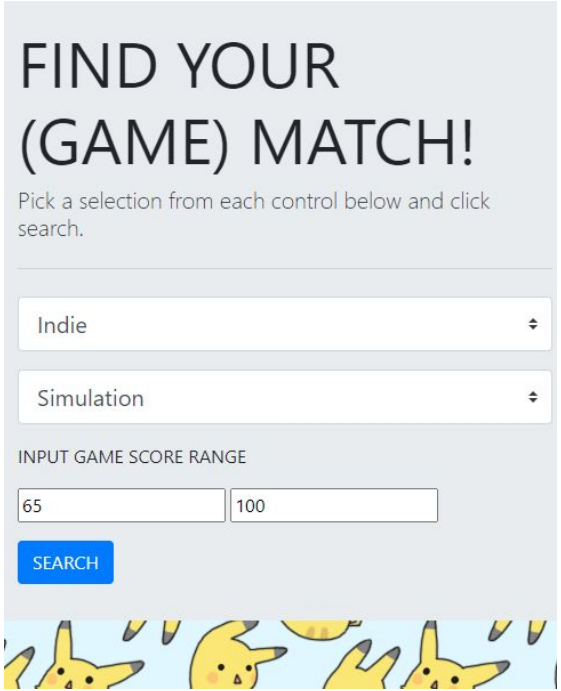
Recommends games based on your inputs of tags, genre and game score

Flask app that queries our database

Sends POST data from fields to py.app which creates varibles for a dynamic query and returns data via http route.

Future Improvements:

- Add machine learning for recommending upcoming games
- Add additional tags and genres to search with (more options)



Http form data management

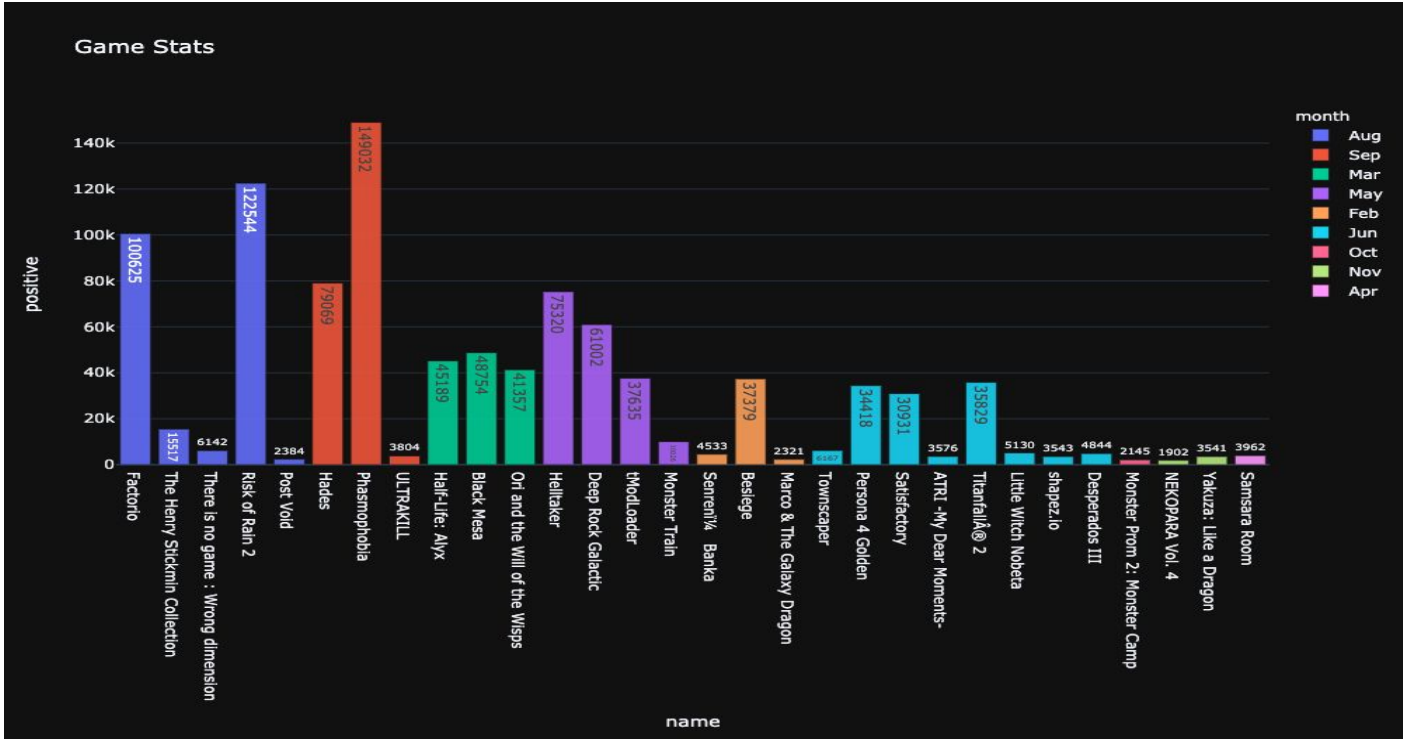
```python
engine=create_engine("sqlite:///assets/data/steam_games.db")

app=Flask(__name__)
CORS(app)

@app.route("/query", methods=["GET", "POST"])
#function to run with POST data trigger from submit form
def query():

    if request.method == "POST":
        req = request.form
        genre = request.form.get("genre")
        tag = request.form.get("tag")
        low = request.form.get("low")
        high= request.form.get("high")
        my_query ="SELECT * FROM games_with_score WHERE genre = "
        my_query+=genre
        results=engine.execute(my_query).fetchmany(50)
        returned_results=[list(result) for result in results]
        return jsonify(returned_results)

    return current_app.send_static_file('query.html')
```

```python
import psycopg2
import collections
import simplejson as json

connection = psycopg2.connect(conn)
cursor = connection.cursor()

cursor.execute("SELECT * FROM steam_clean")
rows = cursor.fetchall()
print(len(rows))
game_list = []
for row in rows:
    d = collections.OrderedDict()
    d['appID'] = row[0]
    d['game_name'] = row[1]
    d['game_score'] = row[2]
    d['genre'] = row[3]
    d['tag'] = row[4]
    d['price'] = row[5]
    d['min_owners'] = row[6]
    d['max_owners'] = row[7]
    d['hit'] = row[8]
    game_list.append(d)

json_object = json.dumps(game_list, indent = 4)
#print(json_object)

with open("sample.json", "w") as outfile:
    json.dump(json_object, outfile)

cursor.close()
connection.close()
print("PostgreSQL connection is closed")
```

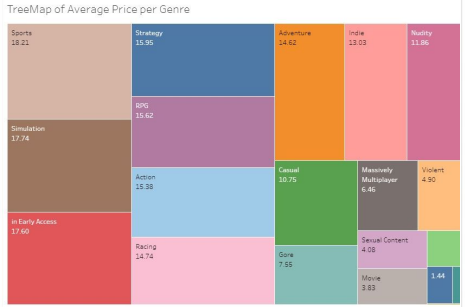

```html
<body background="pika.jpg">
  <div class="jumbotron">
    <h1 class="display-4 text-left"><strong>FIND YOUR (GAME) MATCH!
    </strong></h1>
    <blockquote>
      <h2 class="lead text-left">Pick a selection from each control
      and click search.</h2>
    </blockquote>
    <hr class="my-4">
    <p></p>
    <p class="lead">
    <form action="/query" method="POST">
      <div class="form-group">
        <select class="custom-select custom-select-lg mb-3" id="genr
        name="genre">
          <option selected>Pick Genre of Game</option>
          <option value="Action">Action</option>
          <option value="Strategy">RPG</option>
          <option value="Indie">Indie</option>
          <option value="FPS">FPS</option>
          <option value="RPG">RPG</option>
        </select>
        <select class="custom-select custom-select-lg mb-3" id="tag"
        name="tag">
          <option selected>Pick Tag of Game</option>
          <option value="First-Person">1st Person</option>
          <option value="Single-Player">Single Player</option>
          <option value="Simulation">Simulation</option>
          <option value="Strategyn">Strategy</option>
          <option value="Indie">Indie</option>
        </select>
          <p class="text-uppercase">input GAME score range</p>
        <input name="low" id="low" type="number" value="50"/>
        <input name="high" id="high" type="number" value="100"/>
      </div>
      <button type="submit" class="btn btn-primary">SEARCH</button>
    </form>
```

Plotly

```python
def get_rating_info():

    url = "https://steamdb.info/stats/gameratings/?year=2020"
    headers = {'User-Agent': 'curl/7.65.3'}
    page = requests.get(url, headers=headers)
    soup = BeautifulSoup(page.text)
    game_table = soup.find(id='table-apps')

    return_array = []

    # skip the first TR -> [1:]
    for i in game_table.find_all('tr')[1:]:
        #print(i)
        td = i.find_all('td')
        name = td[2].text.strip(":")
        release = td[3].text
        peak = td[4].text.replace(",", "")
        positive = td[5].text.replace(",", "")
        negative = td[6].text.replace(",", "")

        rating = td[7].text.strip("%")

        return_array.append(
        {'name': name,
         'release': release,
         'peak':peak,
         'positive': positive,
         'negative': negative,
         'rating': rating}
        )
```



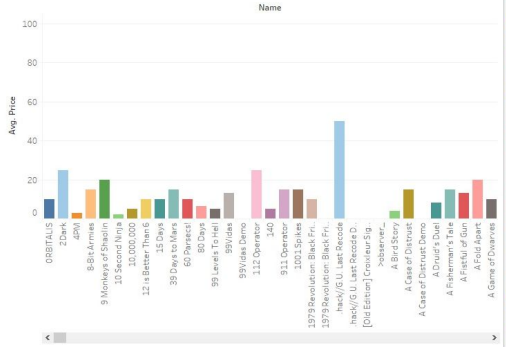
Tableau


TreeMap of Average Price per Genre
Sports 18.21
Strategy 15.95
Adventure 14.62
Indie 13.03
Nudity 11.86
Simulation 17.74
RPG 15.62
Action 15.38
Casual 10.75
Massively Multiplayer 6.46
Violent 4.90
in Early Access 17.60
Racing 14.74
Sexual Content 4.08
Gore 7.55
Movie 3.83
1.44

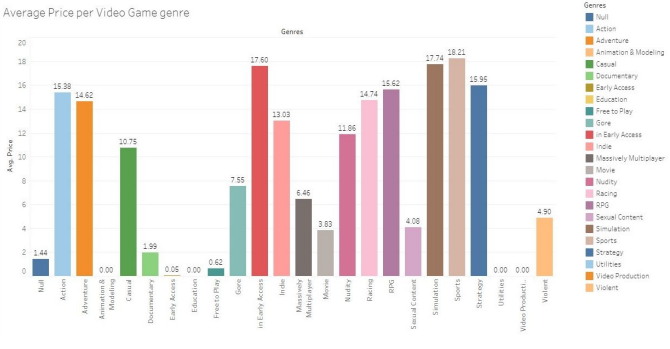

TreeMap of Average Price per game tag
Auto Battler


Word Cloud for Vidoe game genres
RPG
Indie
Racing
in Early Access
Sports
Casual
Adventure
Free to Play
Simulation
Strategy
Action
Massively Multiplayer
Genres
Null
Action
Adventure
Animation & Modeling
Casual
Documentary
Early Access
Education
Free to Play
Gore
in Early Access
Indie
Massively Multiplayer
Movie
Nudity
Racing
RPG
Sexual Content
Simulation
Sports
Strategy
Utilities
Video Production
Violent


Word Cloud of Video Game Tags
Action
Multiplayer
Singleplayer
Adventure
Indie
Simulation
Strategy
Casual
Great Soundtrack
Atmospheric
RPG
Puzzle
Platformer
Story Rich
Shooter
Horror
Sci-fi
First-Person
Difficult
Open World
Fantasy
Female Protagonist
Co-op
Retro

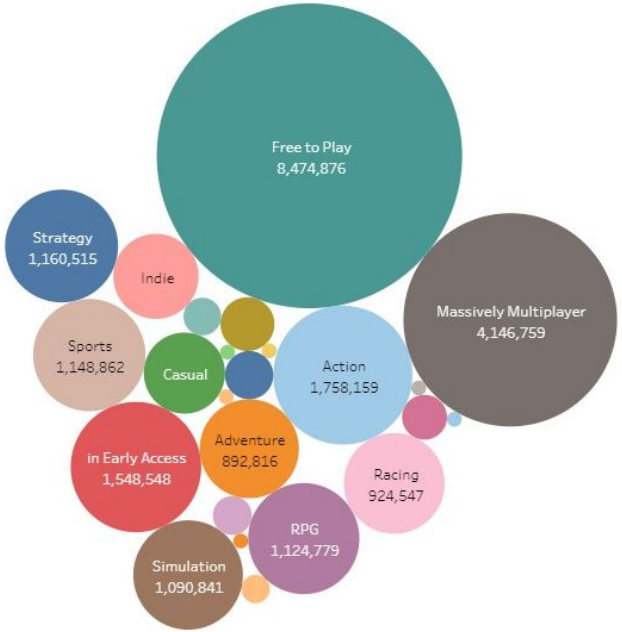

Average Price per game (title)
Name
Avg. Price
100
80
60
40
20
0
0RBITALIS
2Dark
4PM
8-Bit Armies
9 Monkeys of Sha...
10 Second Ninja
10,000,000
12 is Better Than 6
15 Days
39 Days to Mars
60 Parsecs!
80 Days
99 Levels To Hell
99Vidas
99Vidas Demo
112 Operator
140
911 Operator
1001 Spikes
1979 Revolution: Black Fri...
1979 Revolution: Black Fri...
.hack//G.U. Last Recode
.hack//G.U. Last Recode D...
[Old Edition] Crowley's Sig...
>observer_
A Bird Story
A Case of Distrust
A Case of Distrust Demo
A Druid's Duel
A Fisherman's Tale
A Fistful of Gun
A Fold Apart
A Game of Dwarves


Name
0RBITALIS
2Dark
4PM
8-Bit Armies
9 Monkeys of Sha...
10 Second Ninja
10,000,000
12 is Better Than 6
15 Days
39 Days to Mars
60 Parsecs!
80 Days
99 Levels To Hell
99Vidas
99Vidas Demo
112 Operator
140
911 Operator
1001 Spikes
1979 Revolution:...
1979 Revolution:...
.hack//G.U. Last R...
.hack//G.U. Last R...
[Old Edition] Croi...
>observer_
A Bird Story

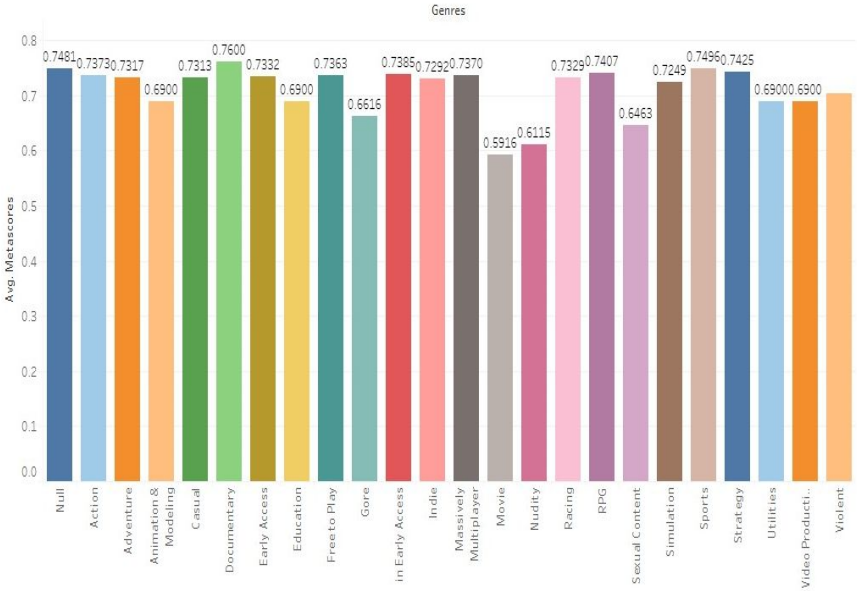

Average Price per Video Game genre
Genres
Avg. Price
20
18
16
14
12
10
8
6
4
2
0
1.44
15.38
14.62
0.00
10.75
1.99
0.05
0.00
0.62
7.55
17.60
13.03
6.46
3.83
11.86
14.74
15.62
4.08
17.74
18.21
15.95
0.00
0.00
4.90
Null
Action
Adventure
Animation & Modeling
Casual
Documentary
Early Access
Education
Free to Play
Gore
in Early Access
Indie
Massively Multiplayer
Movie
Nudity
Racing
RPG
Sexual Content
Simulation
Sports
Strategy
Utilities
Video Production
Violent

Tableau

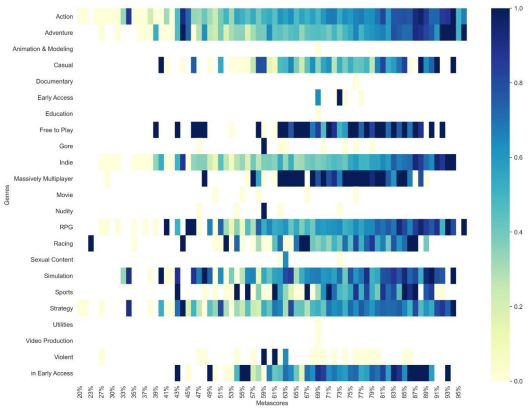
Average number of Max Owners per video game genre



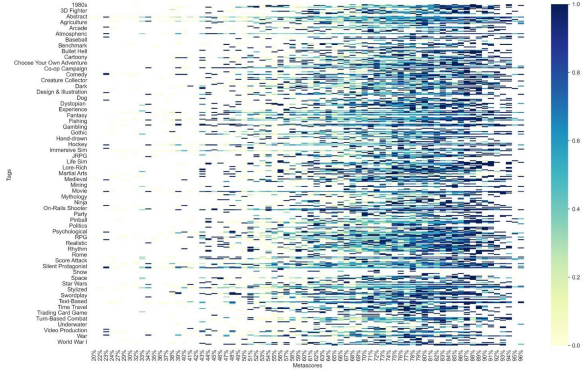
Metascore per Video Game Genre



Other Visualizations



Heatmap figure 1:
This figure compares Genres of video games to their metascore rating. The value is whether the game was a hit (1.0) or not a hit (0.0) based on the number of games owned.



Heatmap Figure 2:
This figure compares game Tags of video games to their metascore rating. Game tags are user inputed and are words to describe games The value is whether the game was a hit (1.0) or not a hit (0.0) based on the number of games owned.

These heatmaps show a trend that typically across most genres and tags the higher the Metascore rating the more likely the game will be a hit. Metascores for games are pulled from the review site Metacritic. Metascore is created from a weighted average of scores given by a critic. A game is labeled as a "Must-Play" when the Metascore is higher than 90 and has been reviewed by at least 15 publications.

Applications of our findings:

- Game developers can use our app to predict how popular their game will be
- Game companies can use our analysis to determine which types of games are most successful
- Gamers can filter through our results to find games similar to ones they already enjoy

Future Questions:

- How does the platform a game is released on determine its' popularity?
- Does the size of the developing company impact the success of the game such as large developers vs. indie developers?
- How does user inputs such as reviews impact game ownership and popularity?

Big Thank You and Shout Outs

- Kevin Lee
- Jess Tillis
- Royal Taylor

MARIO
122100

×26

WORLD
5-4

TIME

GAME OVER