

# 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"
}
```

← → ↻ 🏠 🤖 steamspy.com

Connection failed: Too many connections



**Steam Spy** @Steam\_Spy · Nov 16  
Sorry, the DB is down, investigating.

💬 3



❤️ 8



	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 Tig	
95242	1474960	Der Geisterturn / 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	0Ã .Ã 20,000	[ ]	NaN	\$0.00
1	8	winnu2	NaN	NaN	NaN	0Ã .Ã 20,000	[ ]	NaN	\$0.00
2	10	Counter-Strike	Valve	Valve	88%	10,000,000Ã .Ã 20,000,000	[Action]	[Action, FPS, Multiplayer, Shooter, C...	\$9.99
3	20	Team Fortress Classic	Valve	Valve	NaN	2,000,000Ã .Ã 5,000,000	[Action]	[Action, FPS, Multiplayer, Classic, H...	\$4.99
4	30	Day of Defeat	Valve	Valve	79%	5,000,000Ã .Ã 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("Ã ..Ã ", n = 1, expand = True)  
steam_data["owners"] = new  
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 x 2 columns

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

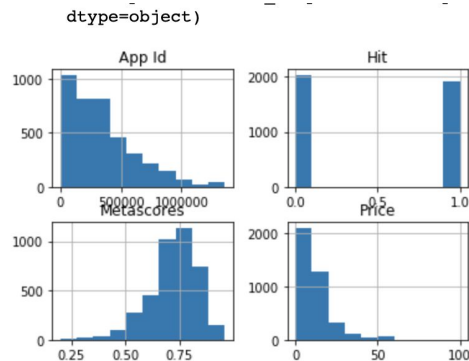
# Our Machine Learning Algorithm

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

```
6 coe_df = coe_df.sort_values(by=['abs'],
7 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

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

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

  fetch("/predict/" + price + "/" + metascoring)
    .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);
    });
}
```

```
# 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>/<metascoring>')
def display(price, metascoring):
    """
    """
    price = float(price)
    metascoring = float(metascoring)
    # user sends metascoring from 1-100
    # but we need 0.0 - 1.0, so divide by 100
    metascoring = metascoring / 100.0

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

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 metascoring, it will predict if the game will be a hit (successful) or not.



Price: \$50



Metascoring: 0.21

```
> {is_hit: 0, metascoring: 0.16, price: 50} (index):135
0 (index):143
> {is_hit: 0, metascoring: 0.17, price: 50} (index):135
0 (index):143
> {is_hit: 0, metascoring: 0.19, price: 50} (index):135
0 (index):143
> {is_hit: 0, metascoring: 0.21, price: 50} (index):135
0 (index):143
> {is_hit: 0, metascoring: 0.23, price: 50} (index):135
0 (index):143
> {is_hit: 0, metascoring: 0.24, price: 50} (index):135
0 (index):143
> {is_hit: 0, metascoring: 0.25, price: 50} (index):135
0 (index):143
> {is_hit: 0, metascoring: 0.24, price: 50} (index):135
0 (index):143
> {is_hit: 0, metascoring: 0.23, price: 50} (index):135
0 (index):143
> {is_hit: 0, metascoring: 0.22, price: 50} (index):135
0 (index):143
> {is_hit: 0, metascoring: 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 deployed through Heroku

Future Improvements:

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

## FIND YOUR (GAME) MATCH!

Pick a selection from each control below and click search.

Pick Genre of Game



Pick Tag of Game



INPUT GAME SCORE RANGE

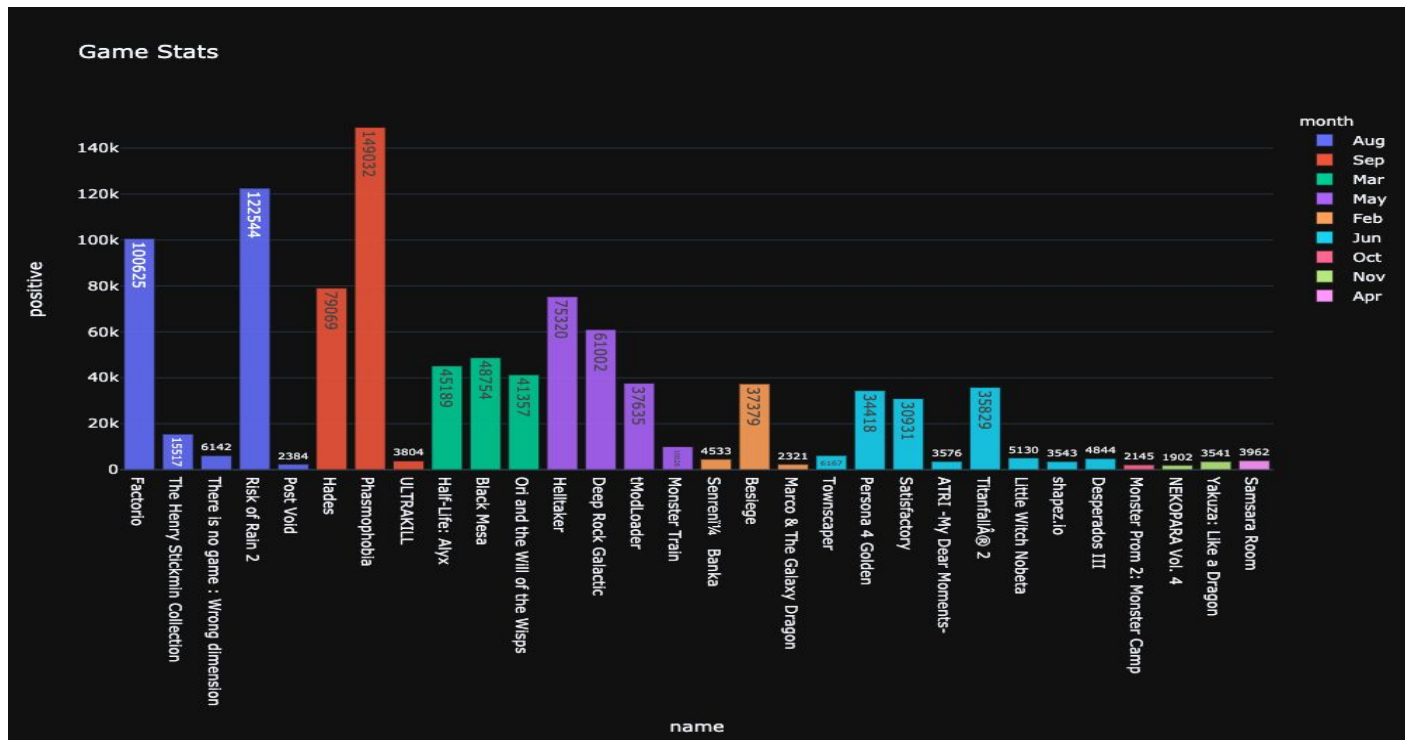
50

100

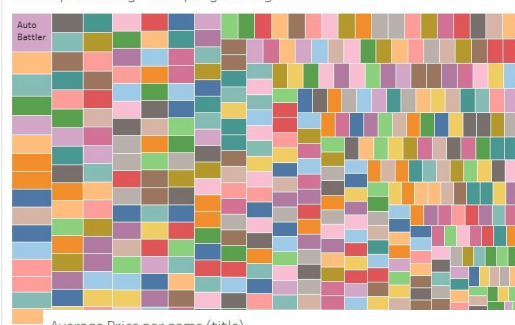
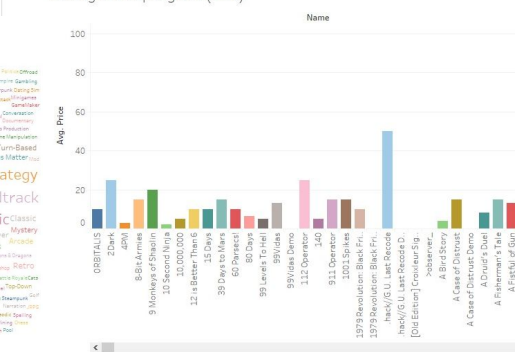
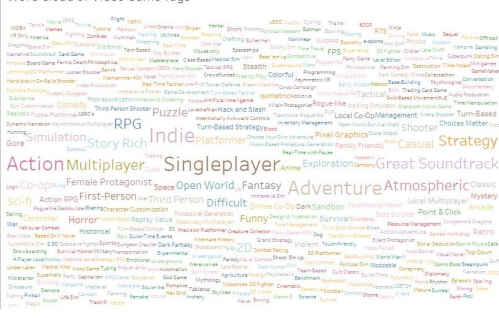
SEARCH



# Plotly



Sports 18.21	Strategy 15.90	Adventure 14.62	Indie 13.03	Reality 11.96
	RPG 15.62			
Simulation 17.74	Action 16.38	Casual 10.75	Massively Multiplayer 6.45	Violent 4.90
in Early Access 17.60	Racing 14.74	Drama 7.55	Sexual Content 4.08	1.45
			Mystery 3.83	

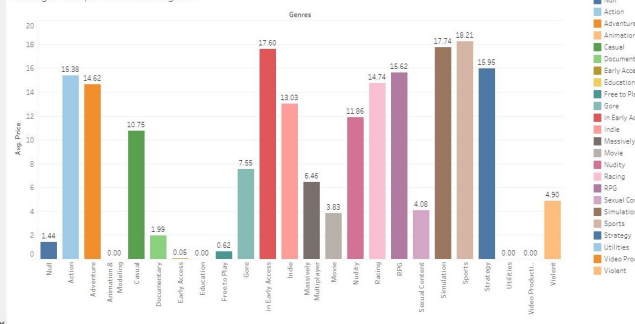
[illegible][illegible]

**Genres**

Genre	Avg. Age
Null	1.44
Action	15.39
Adventure	14.62
Animation	0.00
Comedy	10.76
Cosplay	1.99
Documentary	0.06
Early Access	0.00
Education	0.42
Free to Play	7.55
Game	17.80
Indie	13.09
Massively Multiplayer	6.45
RPG	3.93
Strategy	11.86
Visual Novel	14.74
VR	15.62
Visual Effects	4.08
Simulation	17.74
Sports	18.21
Strategy	15.95
Utilities	0.00
Video Production	0.00
Violent	4.90

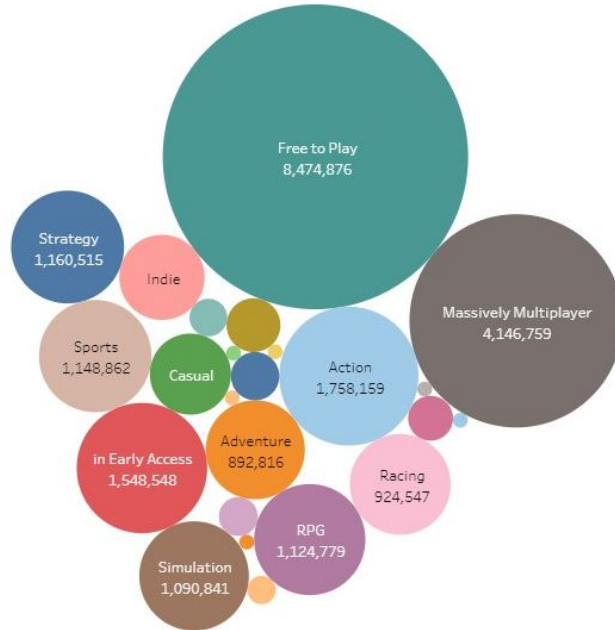
**Legend:**

- Action
- Adventure
- Animation
- Comedy
- Cosplay
- Documentary
- Early Access
- Education
- Free to Play
- Game
- Indie
- Massively Multiplayer
- RPG
- Strategy
- Visual Novel
- Visual Effects
- Simulation
- Sports
- Strategy
- Utilities
- Video Production
- Violent

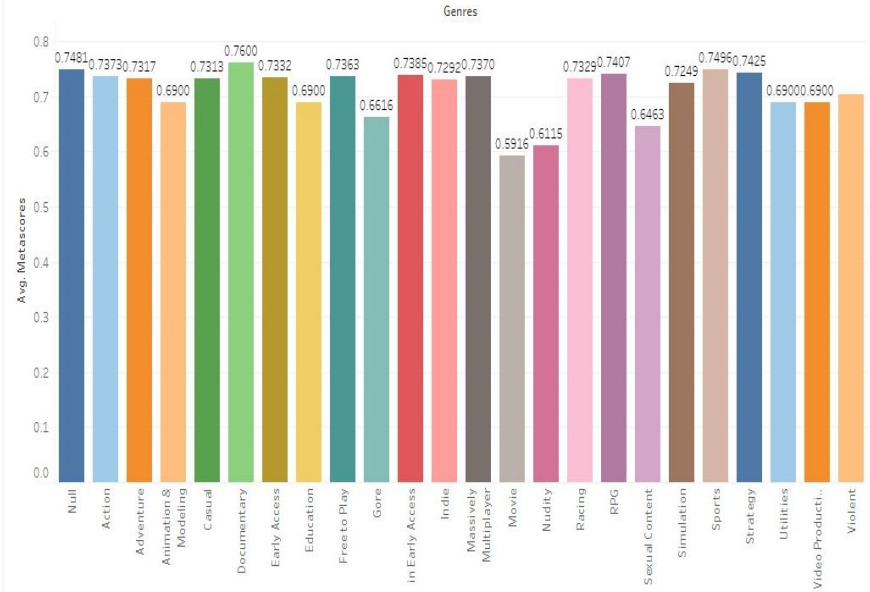


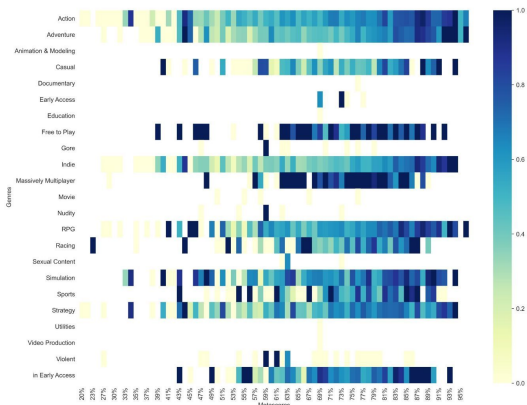
# Tableau

Average number of Max Owners per video game genre



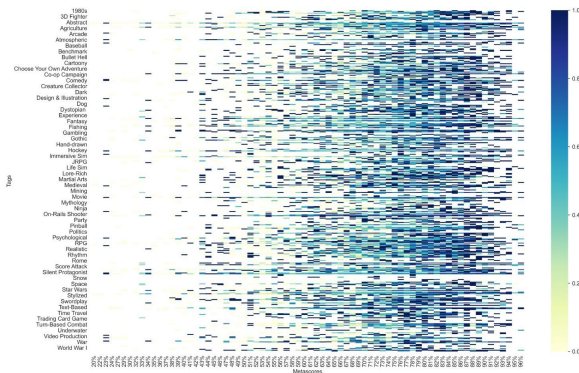
Metascore per Video Game Genre





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 inputted 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

🍄x26

WORLD  
5-4

TIME

GAME OVER