GamingReviewsML

June 20, 2024

Steam Gaming Reviews ML Model

Introduction

In this project the main objective is to figure out what kind of game you should develop on Steam if you are looking to maximize your positive reivew distribution. This would you give you a feel for what genre you should make your game or which features to include if you want to have a game that is well received. To do this, we are analyzing a dataset that contains both features of the games as well as the genre of the game.

ML Method

In this notebook I took the approach of converting the categorical variables into binary flags and doing a Random Forest Regrssion algorithm to see how important each of the cateogical attributes is to maximizing the review score.

[1]:
```
# Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
```

Data Source

The file used in this project was downloaded from the path below. Aman Barthwal. (2024). Steam Store Data. Kaggle.com. https://www.kaggle.com/datasets/amanbarthwal/steam-store-data

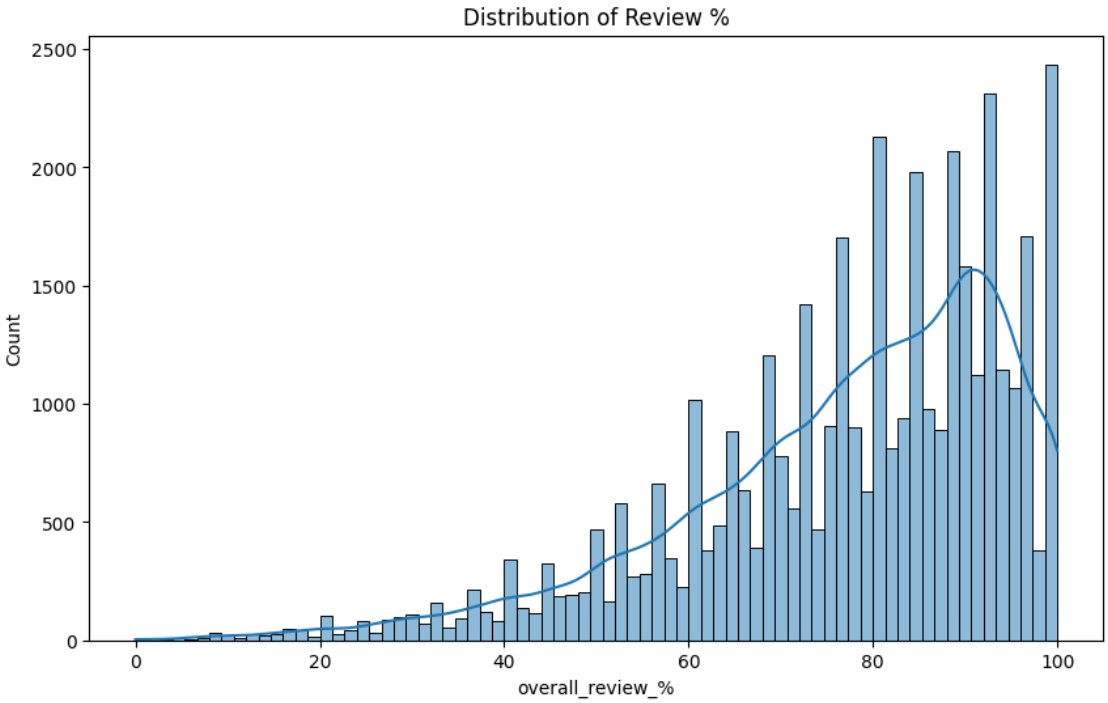
This dataset is a collection of reviews and game meta data from games that are on Steam. The data includes 83,876 rows and 24 columns. The main columns of interest today are comma seperated values that we are going to clean up and split into their only binary flag columns.

[2]:
```
## Loading the data.
df = pd.read_csv('steam-games.csv')
```

EDA

Doing some exploring of the features to get a feel for the data and what the general distribution of the data is.

[3]:
```
#Plot the distribution of the reivew % to get a feel for how reviews are␣
↪distributed.
plt.figure(figsize=(10, 6))
sns.histplot(df['overall_review_%'], kde=True)
plt.title('Distribution of Review %')
plt.show()
```



Data Cleaning

The data set that was chosen is mostly clean. One thing that did need to be done to some columns throughout this process was to replace null values with "Unknown" and to split out comma seperated string fields into their own columns for proper use in the functions.

[4]:
```
#In this section we are splitting the values that are comma seperated fields␣
↪into their own columns with binary flags.

from sklearn.preprocessing import MultiLabelBinarizer

# Fill the NA values with Unknown so that they can process appropriately.
df['genres'] = df['genres'].fillna('Unknown')
df['categories'] = df['categories'].fillna('Unknown')
df['content_descriptor'] = df['content_descriptor'].fillna('Unknown')

# One-hot encode the genres
mlb = MultiLabelBinarizer()
```

```
genre_encoded = mlb.fit_transform(df['genres'].str.split(","))

# Create a DataFrame with the encoded genres
genre_df = pd.DataFrame(genre_encoded, columns=mlb.classes_)

# One-hot encode the categories
mlb = MultiLabelBinarizer()
categories_encoded = mlb.fit_transform(df['categories'].str.split(","))

# Create a DataFrame with the encoded genres
category_df = pd.DataFrame(categories_encoded, columns=mlb.classes_)
```

[5]:
```
# Combine the encoded genres with the original DataFrame
df = pd.concat([df.drop(['genres', 'categories'], axis=1), genre_df,␣
 ↪category_df], axis=1)
```

Machine Learning Models

For both sets of the features the, content and the genres, I performed a Random Forest Regression to get a feel for how the features are affecting the review percantage of the games.

[6]:
```
# Split the dataset into training and testing sets
df['overall_review_%'] = df['overall_review_%'].fillna(0)

X = df[list(genre_df.columns)]
y = df['overall_review_%']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[7]:
```
# Model Selection and Training
model = RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)
```

[7]:
```
RandomForestRegressor(random_state=42)
```

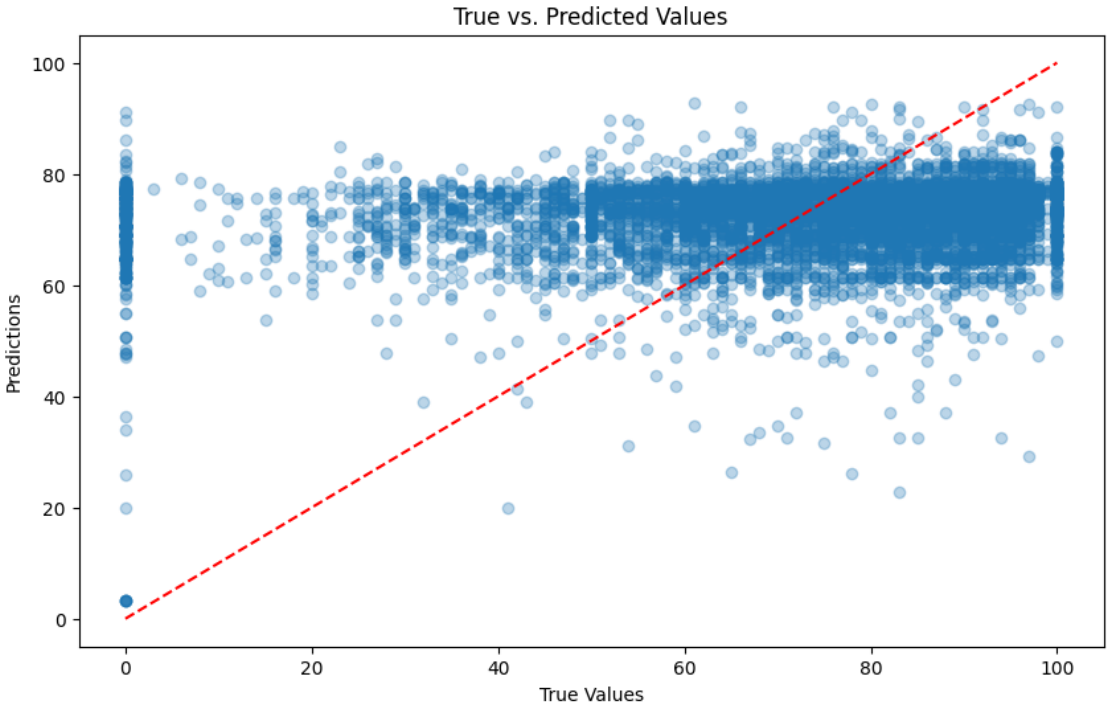
[8]:
```
# Model Evaluation
y_pred = model.predict(X_test)
```

[9]:
```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 595.4816664447793
```

[10]:
```
# Plot true vs. predicted values
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.3)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--r')
plt.xlabel('True Values')
```

```
plt.ylabel('Predictions')
plt.title('True vs. Predicted Values')
plt.show()
```

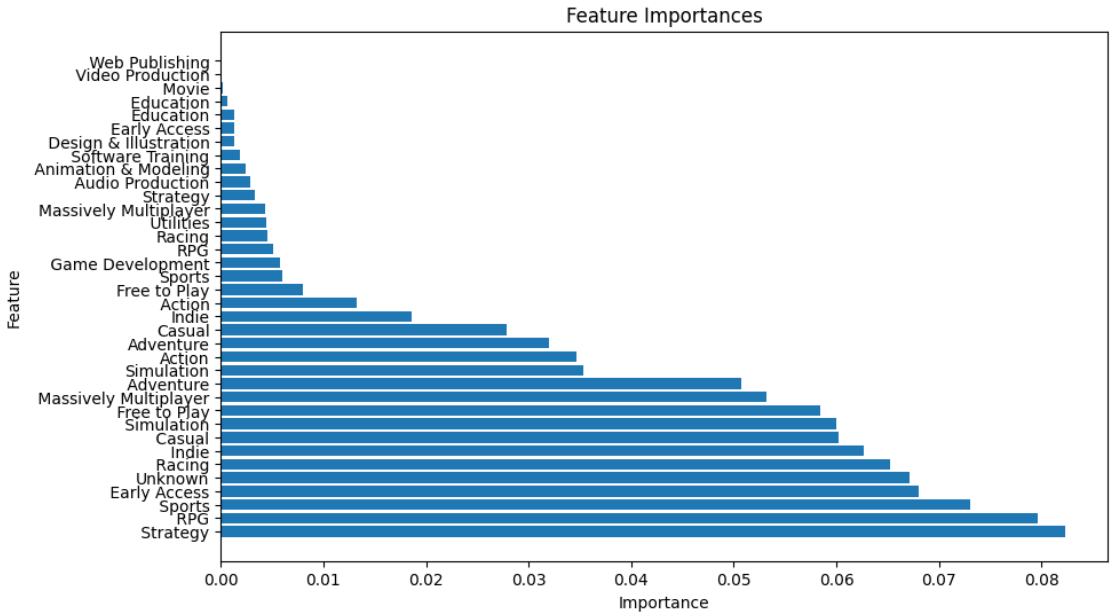


[11]:
```
# Get feature importances
importances = model.feature_importances_
feature_importances = pd.DataFrame({'Feature': X.columns, 'Importance':␣
 ↪importances})
feature_importances = feature_importances.sort_values(by='Importance',␣
 ↪ascending=False)
print("\nFeature Importances:")
print(feature_importances)

# Plot the feature importances
plt.figure(figsize=(10, 6))
plt.barh(feature_importances['Feature'], feature_importances['Importance'])
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.title('Feature Importances')
plt.show()
```

```
Feature Importances:
                    Feature     Importance
```

18	Strategy	8.234572e-02
13	RPG	7.962954e-02
17	Sports	7.299290e-02
6	Early Access	6.807538e-02
36	Unknown	6.711217e-02
14	Racing	6.519300e-02
10	Indie	6.270938e-02
4	Casual	6.023187e-02
15	Simulation	5.999151e-02
8	Free to Play	5.841159e-02
11	Massively Multiplayer	5.323238e-02
1	Adventure	5.076375e-02
32	Simulation	3.538127e-02
22	Action	3.464513e-02
23	Adventure	3.203115e-02
24	Casual	2.788070e-02
28	Indie	1.865474e-02
0	Action	1.326014e-02
27	Free to Play	8.026813e-03
33	Sports	6.017441e-03
9	Game Development	5.779557e-03
30	RPG	5.071415e-03
31	Racing	4.525733e-03
19	Utilities	4.418918e-03
29	Massively Multiplayer	4.372329e-03
35	Unknown	4.124771e-03
34	Strategy	3.324549e-03
3	Audio Production	2.861738e-03
2	Animation & Modeling	2.383062e-03
16	Software Training	1.851347e-03
5	Design & Illustration	1.295715e-03
25	Early Access	1.295275e-03
26	Education	1.271460e-03
7	Education	5.876757e-04
12	Movie	1.865624e-04
20	Video Production	6.287387e-05
21	Web Publishing	4.474433e-07



[12]:
```
# Split the dataset into training and testing sets

X = df[list(category_df.columns)]
y = df['overall_review_%']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[13]:
```
# Model Selection and Training
model = RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)
```

[13]:
```
RandomForestRegressor(random_state=42)
```

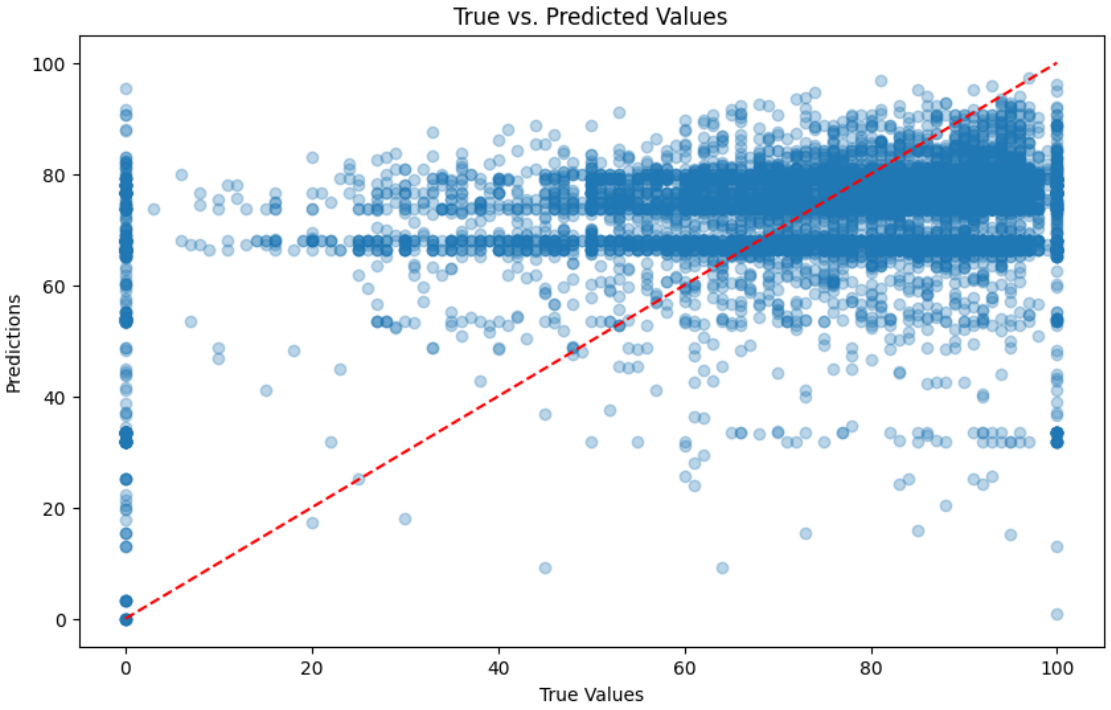
[14]:
```
# Model Evaluation
y_pred = model.predict(X_test)
```

[15]:
```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 546.3952586576065
```

[16]:
```
# Plot true vs. predicted values
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.3)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--r')
plt.xlabel('True Values')
```

```
plt.ylabel('Predictions')
plt.title('True vs. Predicted Values')
plt.show()
```



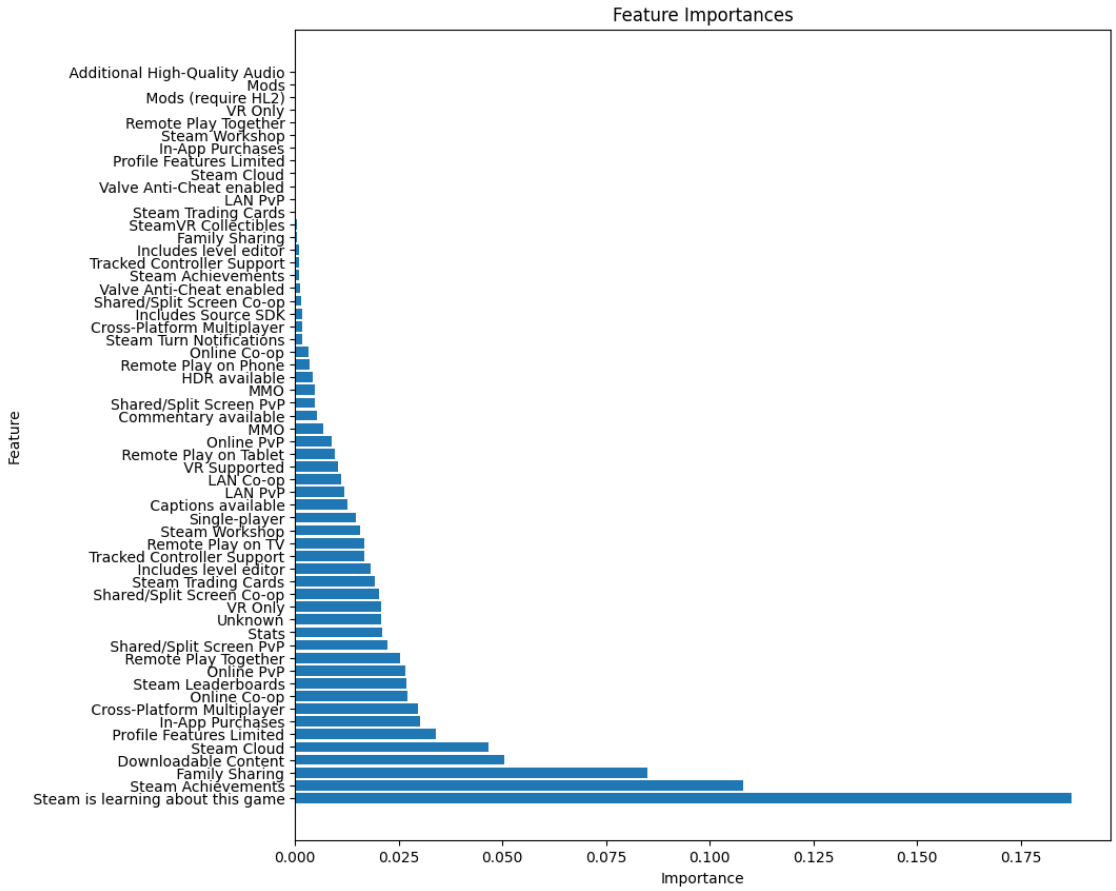
[17]:
```
# Get feature importances
importances = model.feature_importances_
feature_importances = pd.DataFrame({'Feature': X.columns, 'Importance':␣
 ↪importances})
feature_importances = feature_importances.sort_values(by='Importance',␣
 ↪ascending=False)
print("\nFeature Importances:")
print(feature_importances)

# Plot the feature importances
plt.figure(figsize=(10, 10))
plt.barh(feature_importances['Feature'], feature_importances['Importance'])
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.title('Feature Importances')
plt.show()
```

```
Feature Importances:
                                    Feature     Importance
```

31	Steam is learning about this game	1.873139e-01
25	Steam Achievements	1.081291e-01
5	Family Sharing	8.490744e-02
4	Downloadable Content	5.058078e-02
26	Steam Cloud	4.670767e-02
17	Profile Features Limited	3.402921e-02
7	In-App Purchases	3.006439e-02
3	Cross-Platform Multiplayer	2.975191e-02
15	Online Co-op	2.722992e-02
27	Steam Leaderboards	2.687589e-02
16	Online PvP	2.653824e-02
18	Remote Play Together	2.522616e-02
23	Shared/Split Screen PvP	2.227105e-02
24	Stats	2.113382e-02
56	Unknown	2.077021e-02
34	VR Only	2.067697e-02
22	Shared/Split Screen Co-op	2.025789e-02
28	Steam Trading Cards	1.932853e-02
9	Includes level editor	1.831008e-02
33	Tracked Controller Support	1.675000e-02
20	Remote Play on TV	1.668847e-02
30	Steam Workshop	1.564474e-02
49	Single-player	1.455952e-02
1	Captions available	1.257368e-02
11	LAN PvP	1.199286e-02
10	LAN Co-op	1.114385e-02
35	VR Supported	1.033684e-02
21	Remote Play on Tablet	9.556013e-03
44	Online PvP	8.919448e-03
12	MMO	6.758088e-03
55	Unknown	6.581899e-03
2	Commentary available	5.260943e-03
48	Shared/Split Screen PvP	4.793697e-03
42	MMO	4.726199e-03
6	HDR available	4.332765e-03
19	Remote Play on Phone	3.462849e-03
43	Online Co-op	3.220313e-03
29	Steam Turn Notifications	1.820695e-03
37	Cross-Platform Multiplayer	1.704606e-03
8	Includes Source SDK	1.665499e-03
47	Shared/Split Screen Co-op	1.490022e-03
36	Valve Anti-Cheat enabled	1.326869e-03
50	Steam Achievements	9.624160e-04
54	Tracked Controller Support	9.088594e-04
40	Includes level editor	8.904311e-04
38	Family Sharing	5.227492e-04
32	SteamVR Collectibles	3.728590e-04
52	Steam Trading Cards	2.780708e-04

41	LAN PvP	2.265109e-04
58	Valve Anti-Cheat enabled	1.843812e-04
51	Steam Cloud	6.001623e-05
45	Profile Features Limited	5.840067e-05
39	In-App Purchases	3.330981e-05
53	Steam Workshop	3.315721e-05
46	Remote Play Together	2.851928e-05
57	VR Only	1.604254e-05
14	Mods (require HL2)	1.050103e-05
13	Mods	6.973550e-07
0	Additional High-Quality Audio	0.000000e+00



Conclusion

It appears that the genres and categories alone aren't going to accurately predict the review score of a game, but there are some features that are good to include.

For genres; strategy, RPG, and Sports are the top three most influential genres to the score of a game. There are some more that are close behind as well.

For categories; games that are new, have achievements, and support family sharing tend to influence

the review score of a game the most.

Github Link

https://github.com/shaunboerner/MSDS-Boulder-Projects/tree/main/Supervised%20Machine%20Learning