Module3Assignment17

April 17, 2025

[2]:
```
import json
import numpy as np
import matplotlib.pyplot as plt
```

Module 3, Assignment 17

Part 1: Read the mario data

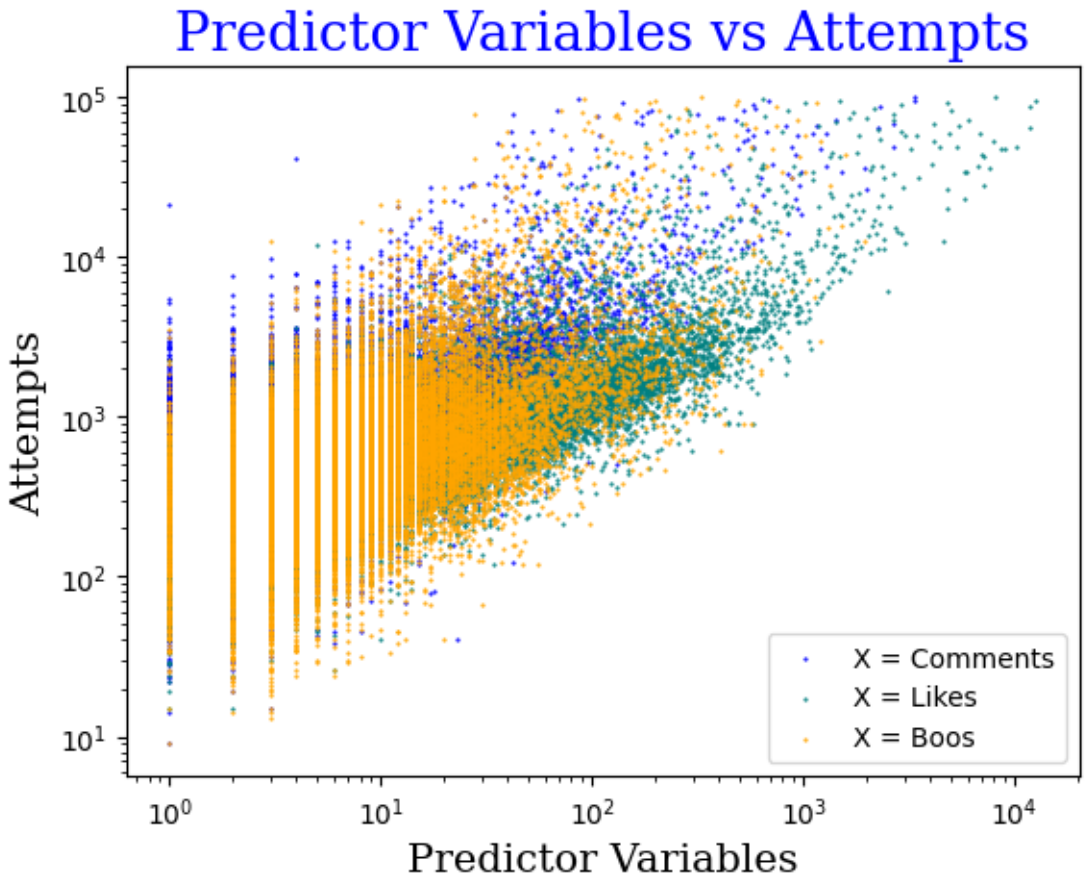
[3]:
```
data_file = "mario_file.json"
with open(data_file, 'r') as mario_file:
    data = json.loads(mario_file.read())

print(f"Number of items: {len(data)}")

# Response
attempts = []

# Predictors
likes = []
boos = []
num_comments = []
for item in data:
    i_attempts, i_likes, i_boos, i_comments = item["attempts"], item["likes"],␣
 ↪item["boos"], item["num_comments"]
    # Restrict each value to be less than upper_bound
    # Otherwise our plot is very visually sparse
    upper_bound = 10**5
    if i_attempts > upper_bound or i_likes > upper_bound or i_boos >␣
 ↪upper_bound or i_comments > upper_bound:
        continue
    attempts.append(i_attempts)
    likes.append(i_likes)
    boos.append(i_boos)
    num_comments.append(i_comments)

attempts = np.array(attempts)
likes = np.array(likes)
boos = np.array(boos)
```

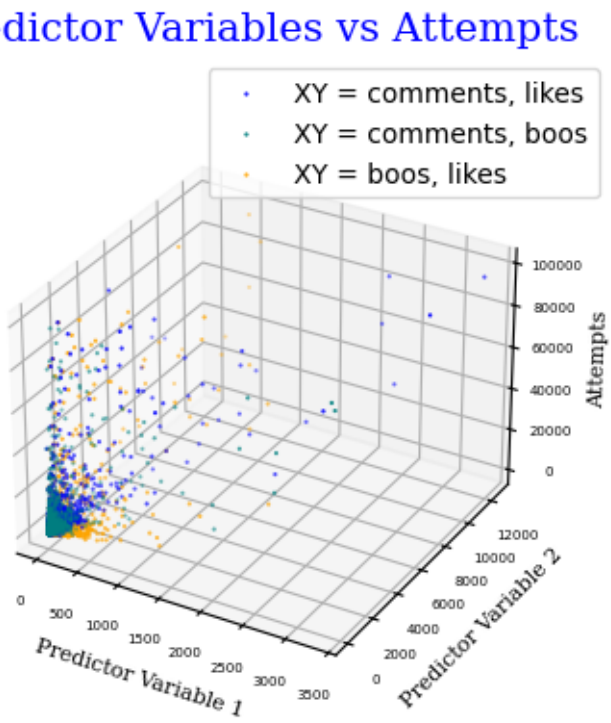
```
num_comments = np.array(num_comments)
```

```
Number of items: 20001
```

Part 2: Plot Mario Data

[34]:
```
font1 = {'family':'serif','color':'black','size':15}
font2 = {'family':'serif','color':'blue','size':20}
plt.xlabel("Predictor Variables", fontdict=font1)
plt.ylabel("Attempts", fontdict=font1)
plt.title("Predictor Variables vs Attempts", fontdict=font2)
dot_size = .3

plt.xscale("log")
plt.yscale("log")
plt.scatter(num_comments, attempts, s=dot_size, color="blue")
plt.scatter(likes, attempts, s=dot_size, color="teal")
plt.scatter(boos, attempts, s=dot_size, color="orange")
plt.legend(("X = Comments", "X = Likes", "X = Boos"))

plt.show()

font3 = {'family':'serif','color':'black','size':8}
font4 = {'family':'serif','color':'blue','size':15}
fig = plt.figure()
ax = fig.add_subplot(projection="3d")
ax.scatter(num_comments, likes, attempts, '^', s=dot_size, color="blue")
ax.scatter(num_comments, boos, attempts, s=dot_size, color="teal")
ax.scatter(boos, likes, attempts, s=dot_size, color="orange")
ax.legend(("XY = comments, likes", "XY = comments, boos", "XY = boos, likes"))
ax.set_xlabel("Predictor Variable 1", fontdict=font3)
ax.set_ylabel("Predictor Variable 2", fontdict=font3)
ax.set_zlabel("Attempts", fontdict=font3)
ax.tick_params(axis='both', which='major', labelsize=5)
ax.set_title("Predictor Variables vs Attempts", fontdict=font4)
ax.set_box_aspect(aspect=None, zoom=0.8)

plt.show()
```


Predictor Variables vs Attempts
Attempts
Predictor Variables
X = Comments
X = Likes
X = Boos



Box 1: Background

We are interested in studying the Mario Maker data because it can give us insights into what types of interactions are most facilitative of player engagement.

This kind of analysis has analogies to social media algorithms, which are largely a complete mystery to the people who use them, though they exert major influence over our lives.

According to Pew Research, "Some 85% of TikTok users say the content on their "For You" page is at least somewhat interesting, including 40% who call it either extremely or very interesting. Only 14% say it is not too or not at all interesting." Most people using these applications are primarily consuming content served to them by the algorithm, so it would be interesting to gain insight into how a social media algorithm may select content for its users.

Box 2: Description of Data

The data I am working with compares comments, likes, and boos against # of total attempts played for a given mario maker level. I have plotted it on a log-log scale, because the amount of data points at any given linear scale decreases in density as the magnitude of the point increases. This results in the graph being extremely sparse if it is plotted linearly.

There appears to be a positive relationship between [comments, likes, boos], and total attempts. Boos appear to be the least visually correlated with attempts, and likes appear to be the MOST visually correlated.

[ ]: