YALE CENTER FOR
Health & Learning Games

Evaluating Character Differences in the Top and Bottom Performing Students with a Custom Score

Aryan, Joey, Lina

Methodology

How can we compare players?

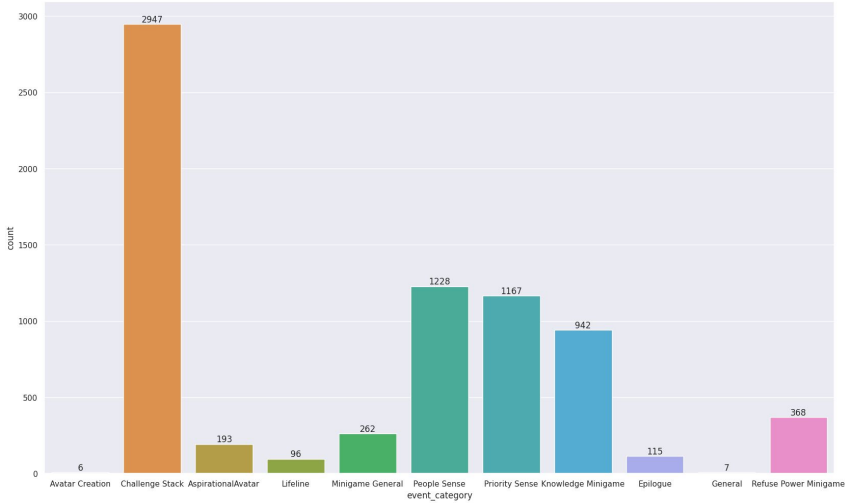
What can we learn about the players?

Challenges

- Survey results not clear
- Large number of extraneous player logs
- Player information hidden

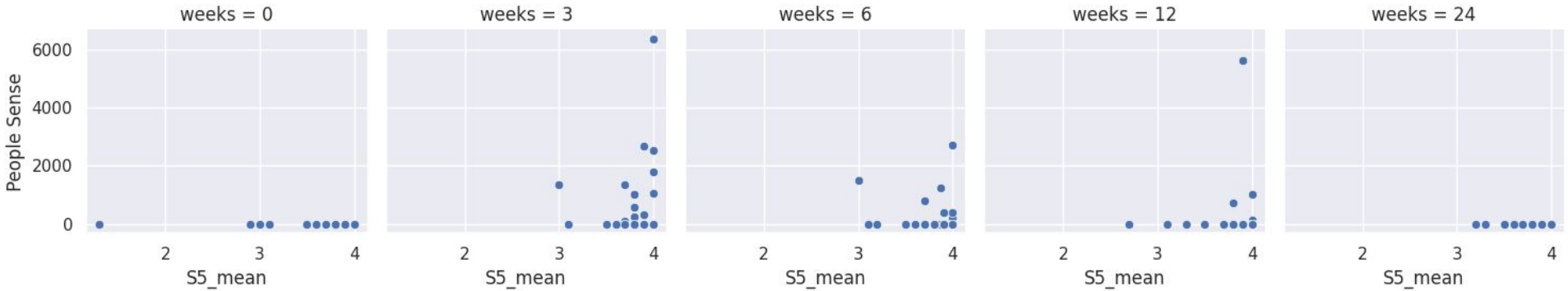
Areas of Interest

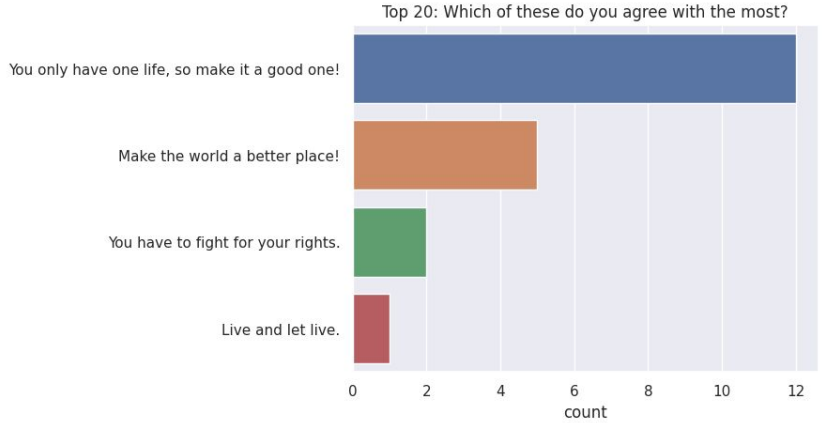
- **“All about me” responses**
- **Play speed and completion**

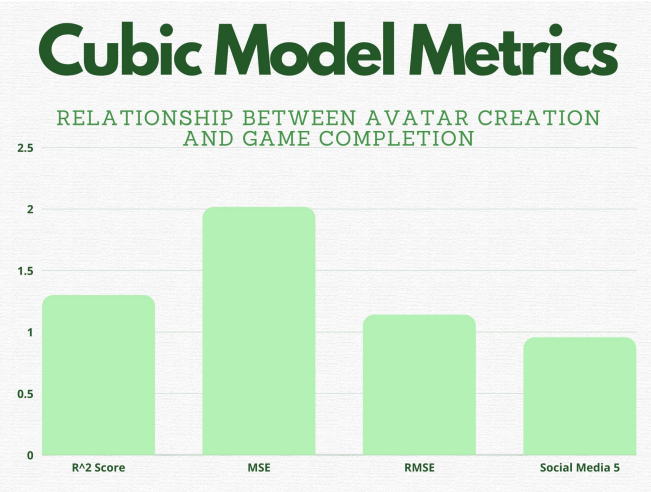


1. Process Aspirational Avatar logs for player information
2. Identify sequence of events
3. Log average time spent on minigames and completion

Modeling and Evaluation







Reflection

Found a meaningful way to compare players

Much left to analyze within data

Major Takeaways

- Students in the higher and lower ranks had slight differences in goals and priorities
- As time went on survey results shifted to the left and showing that the students were better at refusing drugs

Appendix

Score Calculation

```
minigame_results = results['avg_refusal'] + results['avg_people'] + results['avg_priority'] + results['avg_know']
results['avg_minigame'] = minigame_results / 4
results['score'] = (11 - results['max_stack_id']) * 500 + results['avg_minigame']
results['score'] = results['score'].rank(method='min')
```

Linear Regression

```
mydf_final = pd.read_csv('final_data_nums.csv')
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

# countdata = mydf_final.copy()
# countdata['playtime'] = traindata['playtime']
mydf_final = mydf_final.fillna(0).head(50)
# print(mydf_final.shape)
X = mydf_final.values
y = results['score'].head(50).values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)

print_X_test = X
print_y_test = y

from sklearn.preprocessing import PolynomialFeatures
poly_data = PolynomialFeatures(3, include_bias=False).fit_transform(X_train)
poly_data_test = PolynomialFeatures(3, include_bias=False).fit_transform(X_test
# print(poly_data[0:3])

cubic_model = LinearRegression()
cubic_model.fit(X=poly_data, y=y_train)

test_predict = cubic_model.predict(poly_data_test)
```

Scatterplot

```
unique_users = data['player_id'].unique()
new_columns = ["Challenge Stack", "People Sense", "Knowledge Minigame", "Priority Sense", "Refuse Power
for column in new_columns:
    s5_scores[column] = [0]*s5_scores.shape[0]

for user in unique_users:
    one_user = data.loc[data['player_id'] == user]
    date_conversion = one_user.loc[one_user['player_id'] == user, "date"].array
    d0 = datetime.strptime(date_conversion[0], "%Y-%m-%d")

    time_ranges = []
    for week in s5_scores.loc[s5_scores['player_id'] == user]["weeks"]:
        d2 = d0 + timedelta(days=week*7)
        time_ranges.append(d2)

    if not time_ranges:
        continue

    # Convert the date to datetime64
    one_user['date'] = pd.to_datetime(one_user['date'], format='%Y-%m-%d')

    for t in range(1, len(time_ranges)):
        # Filter data between two dates
        start_date = time_ranges[t-1]
        end_date = time_ranges[t]
        filtered_df = one_user.loc[(one_user['date'] >= start_date) & (one_user['date'] < end_date)]
        result = filtered_df["event_category"].value_counts()
        if result.empty:
            continue
        week = ((end_date - d0)/7).days

        for index, value in result.items():
            if not value:
                continue

        for index, value in result.items():
            s5_scores.loc[(s5_scores['player_id'] == user) & (s5_scores['weeks'] == week), index] = val

    #each week has the total values for challenge stack and the score for s5_scores

#make scatterplot

# kmeans = KMeans(n_clusters=3, random_state=0)
# s5_scores['cluster'] = kmeans.fit_predict(s5_scores[['Attack', 'Defense']])
#

sns.relplot(data=s5_scores, x="S5_mean", y="Challenge Stack", col="weeks", height=3)
# sns.relplot(data=s5_scores, x="S5_mean", y="People Sense", col="weeks", height=3)
# sns.relplot(data=s5_scores, x="S5_mean", y="Knowledge Minigame", col="weeks", height=3)
# sns.relplot(data=s5_scores, x="S5_mean", y="Priority Sense", col="weeks", height=3)
```