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

Top 20: Which of these do you agree with the most?

- You only have one life, so make it a good one!
- Make the world a better place!
- You have to fight for your rights.
- Live and let live.

Cubic Model Metrics

Relationship between Avatar Creation and Game Completion

- $R^2$ Score
- MSE
- RMSE
- Social Media 5
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

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

```python
import pandas as pd

# load data
mydf_final = pd.read_csv('final_data_nums.csv')

# import necessary libraries
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

# set random seed for reproducibility
import numpy as np
np.random.seed(0)

# count data
X = mydf_final.values
y = results['score'].head(50).values

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

# fit model
from sklearn.preprocessing import PolynomialFeatures
poly_data = PolynomialFeatures(degree=3, include_bias=False).fit_transform(X_train)
poly_data_test = PolynomialFeatures(degree=3, include_bias=False).fit_transform(X_test)

# fit and predict
linear_model = LinearRegression()
linear_model.fit(X=poly_data, y=y_train)

# predict test values
test_predictions = linear_model.predict(poly_data_test)
```
Scatterplot

```python
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

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

if not time_ranges:
    continue

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

for t in range(0, 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

# For each week, count the number of times a player has scored
for index, value in result.items():
    if value == 0:
        continue

# Each week has the total score for s5_scores

data = s5_scores
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)
```