From data to DESIGN:

Ethical Approaches to Video Game Engagement

Machine Learning and Data Analytics

Jasleem Bonilla
Camille Lojewski
Kimberly Chavez
Adam Tangonan
Diego Hernandez
Saul Topete

Abstract
This study investigates the complex relationship between player engagement and in-game activities to understand how user retention is influenced by game design. The data used is from Grand Theft Auto V Online to identify trends across three months' worth of gameplay data. This study additionally explores the moral implications of game design, including the risk of addiction and the targeting of weaker demographics. Using Lasso, Ridge, and Random Forest regressive processing, the final model was able to obtain 84% prediction accuracy on unseen data. Discussion of our results underlines how important it is to create responsibly designed entertainment and contrast how analytics can both create increased revenue with high engagement, as well as consideration for morally sound gaming experiences.

From Data to Design

1. Introduction

1.1 Motivation

This study focuses on how game design components can increase screen time and possibly user numbers, alongside the morally questionable ease of understanding user behaviors using such all-encompassing data. With the rapid expansion of data analytics in the video game industry, comprehending these dynamics is essential for both financial success and the development of more captivating and enjoyable gaming experiences.

2. Problem description

2.1 Problem statement

Rapid changes in the digital realm have raised industry standards across game quality and consumer expectations, making the gaming industry more competitive. Estimated to be worth over $3 billion, Rockstar Games is one of many gathering extensive data of in-game player behavior. Understanding player preferences and behavior patterns through data analysis can help developers enhance the quality of their games and increase user satisfaction. As Rockstar prepares to release a new game in the series, GTA VI, we analyzed game behaviors to understand how Rockstar may measure player engagement. Developers are constantly challenged to improve design tactics and ensure their games stay profitable, competitive, and satisfactory to fluctuating player expectations by predicting behavioral tendencies.

2.2 Objectives

As well as investigating a metric for competitive edge, this study additionally addresses a gap in the body of knowledge about the moral implications of game design. It is crucial to strike a balance between business goals and the need to refrain from encouraging addictive habits or unfairly singling out vulnerable subpopulations to increase user engagement. As a result, this research is important to both industry stakeholders and the general public since it aims to promote a conversation about responsible gaming behaviors in addition to providing information for better game design. Our analysis is comprehensively motivated by this combined focus on ethical considerations and commercial enhancement, to make a good contribution to the future of gaming.

3. Data

3.1 Data sources

The data is sourced from internal Rockstar Games analytics, comprising three months of users in-game behaviors from Grand Theft Auto Online in fall 2020. The data is categorized into three sets: (1) in-game purchase behavior, (2) online quest activity, and (3) player gaming statistics, including daily playtime logs.

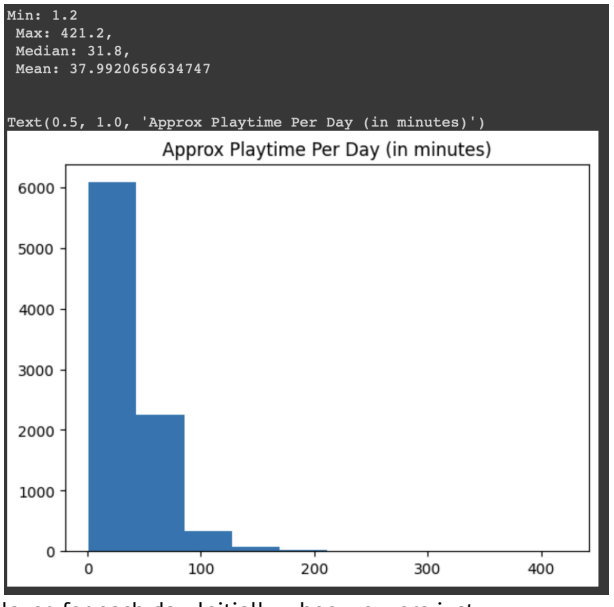
3.2 Data collection and processing

Initial data exploration was structured by Player ID queries of each individual across all datasets, but with a total of 1.25 million rows, the data required more efficient granularity analysis. Early in our analysis, we identified unique Player IDs and occurrence dates that remained consistent across all datasets, providing a solid base for robust cross-sectional and time-series analyses. One standout discovery during exploration was that not every player's screen time was present for all 91 days of data collection, giving us reason to believe the data was pre-processed or truncated by Rockstar. However, each player appeared accurately represented for the days that they were recorded. This required an overhaul of our expected processing method. Our strategy made use of optimization techniques like column-specific processing and selective data querying to manage our large datasets without sacrificing computational effectiveness. Instead of relying on the summed playtimes for each player, we instead considered the ratio of hours each individual played over the number of days that individual was recorded.

After understanding the data's missing components, time granularity, and column representations, we were able to offload data processing into efficiently streamlined functions. We created specialized functions to control the complexity of our dataset. These features were essential for easily separating the data according to several behavioral parameters and getting it ready for analysis. Generally, each dataset was handled as a whole, with small adjustments for features like classifying item types.

3.3 Challenges in data wrangling/handling

Our greatest challenge was handling such a large dataset, as well as incomplete data. A ratio of playtime per documented day had to be developed for each variable. This was due to the incomplete playtime data logging, which necessitated adjustments in the data pipeline. Additionally, from a total count of 9,600 participants, 800 were eliminated since only those with data appearing in all three datasets were suitable for machine learning study. We chose this route to maintain data quality and dependability in later modeling.[1]



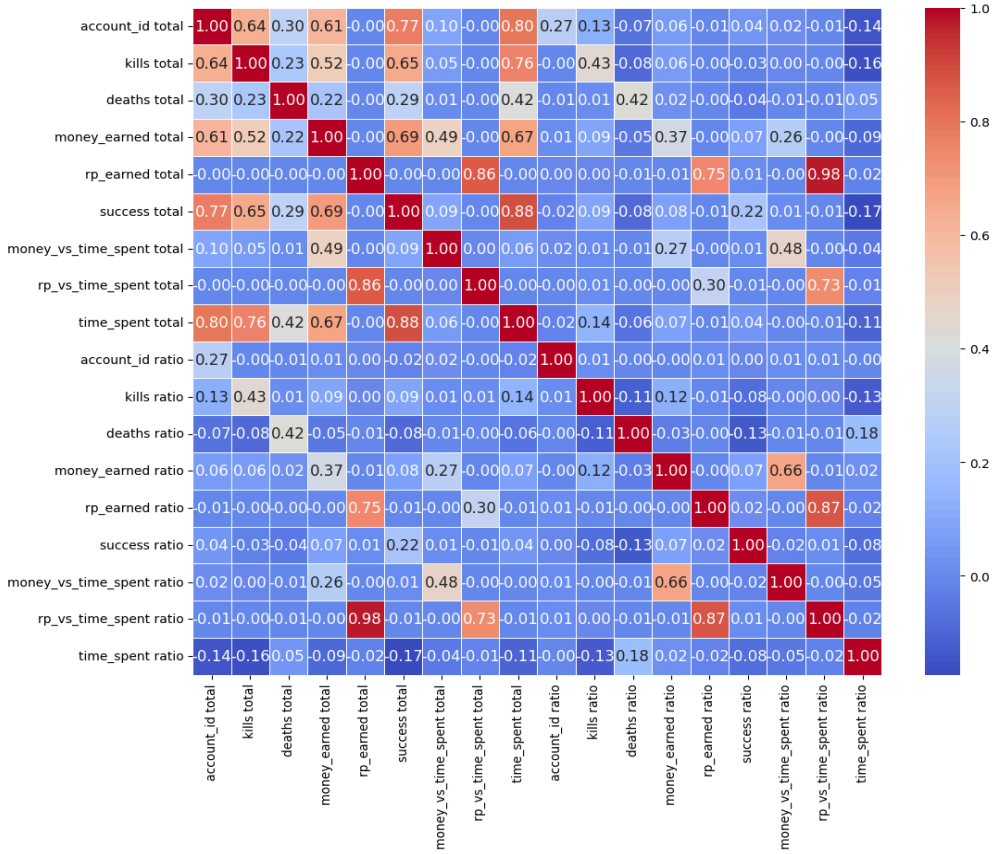
4. Analytic Techniques

4.1 EDA and Selection of Model

A comprehensive exploratory data analysis phase was carried out to explore player behaviors

[1] This was playtime data after finding approximate ratios for each player, for each day. Initially when we were just summing it, the spread of this graph reached 36,000 minutes.

and engagement patterns. The gaming dataset was analyzed using visualizations like heat maps and histograms to look for patterns, correlations, and outliers. We improved our feature selection process by thoroughly examining multicollinearity using correlation heatmaps and the variance inflation factor (VIF), ensuring that the most predictive and independent variables were included in our model. During this phase, the foundation for the following modeling work was established, which helped in identifying important features that have a significant impact on player engagement. Regression and classification algorithms were among the machine learning models that were also used to explore and predict player engagement based on game interaction.[2]



4.2 Implementation details

The initial implementation used a Random Forest model for our data, selected due to its proven capacity to handle outliers and model non-linear interactions, which are essential for deciphering intricate player actions. The chosen features were selected from a heatmap generated after data cleaning and manipulation. The most highly correlated features appeared to be: total kills, the total money, and successes in each player's mission completion. The Random Forest model initially seemed to fit best with our feature selection, as it was robust to the significant noise and outliers present in our large dataset. It was able to capture the non-linear relationships of all 3 features through its aggregation of multiple decision trees, which were valuable in determining to use this model for our feature selection.

4.3 Results and confidence

To create a baseline model for comparison, we used a forgiving and simple approach that made random predictions within a standard deviation of the mean, resulting in an average accuracy of 25%. Our initial model fared better, but without significant predictive power–

[2] Heatmap feature exploration to choose best fit correlated features for our model. Please zoom in to 150% to see details.

achieving only 65% accuracy. Yet after deeper data processing techniques, our final Lasso Regression model was able to significantly improve to consistent 84% accuracy on unseen data.

Key player engagement patterns were more accurately captured by the model thanks to meticulous feature refinement with emphasis on multicollinearity management. In comparison to our Forest Regressor, which ultimately achieved 83% accuracy, our Lasso model was able to better identify nonlinear relationships captured by random playtime data.

4.4 Potential improvements

There is still room for improvement, achievable with deeper feature engineering, but, more importantly, obtaining more consistent data to enable more sophisticated insights. A useful extension might be specific demographic information. This could improve the model's capacity to recognize subpopulations with unique engagement patterns, leading to even more precise and customized forecasts.

Rockstar might be hesitant to give out the analytics they record for a myriad of reasons, for example, the apparent errors and missing values present is our sets. Their data scientists undoubtedly have more precise models with the more complex data they have available.

However, there is room to further improve our selected features, and account for hyperparameter adjustment to improve our model’s accuracy. Deeper analysis might lead us to identify more complex patterns by adjusting the total numbers and depth of our predictive trees. By doing this, the Random Forest model may exceed that of our Lasso model, due to its enhanced capacity to generalize new data.

5. Impact of the Analysis

5.1 Potential and actual impact

Our analysis's main commercial relevance is the possibility of creating games that are both more interesting to play, and possibly more addicting. This realization is crucial for game design as well as for comprehending how video game addictions arise and how businesses may inadvertently or purposely target vulnerable groups of people.

5.2 Ethical considerations and subpopulations

Addiction Risk: Video games' tendency to be addicting can be dangerous, especially for young individuals and those with a history of addiction. The predictive power of this model would ideally be used cautiously to avoid taking advantage of player vulnerabilities.

Targeting and Fairness: Due to cultural, economic, or psychological variables, various subpopulations may respond to the same gaming features in different ways. Although this methodology can assist in customizing game experiences for a wide range of player bases, it is crucial to make sure that these personalizations do not result in the unjust targeting of vulnerable populations. To make sure that player engagement strategies are responsible, complete transparency of existing targeting models would be needed.

Regulation and Monitoring: Part of a larger plan to solve these ethical issues could involve legislation requiring developers to take into account and minimize potential harms in addition to ongoing monitoring of how games affect players.

5.3 Scope for expansion

Although our version of this data was anonymized, it is almost certain Rockstar did not collect this data under the same constraints. More complete data sets, containing player attributes like age and socioeconomic status that were excluded, would greatly expand the currently available area of investigation. In addition to offering insights into typical player preferences and actions, developers could legitimately target or modify experiences to diverse groups by knowing user demographics, but would have to actively avoid methods taking advantage of vulnerable subpopulations.

Adding more online games in the same genre (ie: Fortnite, Red Dead Redemption Online, Roblox, etc.) to the data collection could also yield comparative insights into player preferences and behavior consistency across various gaming platforms and styles. An extended dataset of this kind would ideally allow for a more thorough knowledge of the particular in-game actions that, across a variety of game kinds and player demographics, most substantially drive player engagement and addiction.

Finally, a current dataset that did not occur during the heart of the pandemic could potentially show us how play behaviors have changed over time. This is most notable when considering the most recent updates to popular jobs/activities in GTA Online. From an industry perspective, this would be crucial to understand before the release of GTA VI for game developers to tailor an updated gaming experience.

6. Conclusion

As demonstrated by our model's 84% accuracy in predicting player behavior, our study of the connection between game design and player engagement has produced a valuable discussion regarding the ethics of screentime analytics. By means of rigorous data cleaning, feature selection, and modeling methodologies, we were able to demonstrate the high correlation of game mechanics and player screentime, even through the use of truncated data. Our research adds to the current discourse on ethical game design by supporting approaches that strike a balance between business objectives and player welfare. Greater investigation using larger, more thorough data and varied game genres may improve our comprehension of design and user impact. This is a critical first step in addressing design tactics in the gaming sector.

gta_proj_indeng142_final

May 8, 2024

1 Imports

[ ]:
```
#from google.colab import drive
import pandas as pd
import numpy as np
import random
from pandas import Timestamp
from matplotlib import pyplot as plt

import seaborn as sns
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor

from sklearn import linear_model
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.svm import SVC

from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import StratifiedShuffleSplit
from sklearn import neighbors

from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

from sklearn.ensemble import RandomForestRegressor
```

1.1 Retrieve .csvs:

```
[ ]: ## From GitHub

gta_item_spend_link = 'https://raw.githubusercontent.com/oyrange/GTA-Screentime/
 ↪main/item_spend.csv'
gta_item_spend = pd.read_csv(gta_item_spend_link,
                             parse_dates=['occur_date'])

gta_activity_link = 'https://raw.githubusercontent.com/oyrange/GTA-Screentime/
 ↪main/player_activity.csv'
gta_player_activity = pd.read_csv(gta_activity_link,
                             parse_dates=['occur_date'])

gta_statistics_link = 'https://raw.githubusercontent.com/oyrange/GTA-Screentime/
 ↪main/player_statistics.csv'
gta_player_statistics = pd.read_csv(gta_statistics_link,
                             parse_dates=['occur_date', 'first_day_played'])
```

```
[ ]: len(gta_player_activity) + len(gta_player_statistics) + len(gta_item_spend)
```

```
[ ]: 1247505
```

2 Function and Variable Documentation:

2.0.1 *gta_account_ids and gta_occur_dates represent only ids and dates occurring in all three CSVs:*

```
[ ]: ### Unique `account_id`s appearing in ALL three csvs:

gta_account_ids = np.intersect1d(gta_player_statistics['account_id'],
                                 np.intersect1d(gta_item_spend['account_id'],
                                                ␣
 ↪gta_player_activity['account_id'])
                                 )

print(f'There are {len(gta_account_ids)} `account_id`s appearing in ALL three␣
 ↪\n \
csvs, look them up in gta_account_ids \n')


### Unique `occur_date`s appearing in ALL three csvs:

gta_occur_dates = np.intersect1d(gta_player_statistics['occur_date'],
                                 np.intersect1d(gta_item_spend['occur_date'],
                                                ␣
 ↪gta_player_activity['occur_date'])
```

```
                                    )

print(f'There are {len(gta_occur_dates)} `occur_date`s appearing in ALL three␣
 ↪\n \
csvs, look them up in gta_occur_dates')
```

```
There are 8772 `account_id`s appearing in ALL three
 csvs, look them up in gta_account_ids

There are 91 `occur_date`s appearing in ALL three
 csvs, look them up in gta_occur_dates
```

2.1 Functions:

[ ]:
```
random_state = 8


def process_sums_ratios(df, cols_of_interest, iterations=gta_account_ids):
  """
  Sums and find ratios of cols_of_interest,
    returns them as a df indexed by iterations (default is gta_account_ids).
  """
  totals_array = []
  ratios_array = []
  trimmed_df = df[cols_of_interest]
  for id in iterations:
    ## Only values in df referencing ID.
    id_df = find_subset_in_df('account_id', id, trimmed_df) #dict_df[id]

    ## Totals
    totals = {col+' total': id_df[col].sum()
              for col
              in cols_of_interest}
    totals_array.append(totals)

    ## Ratios
    ratios = {col+' ratio': calculate_listlike_ratio(id_df[col])
            for col
            in cols_of_interest}
    ratios_array.append(ratios)

  ## Join into one DF
  totals_and_ratios_df = pd.DataFrame(totals_array, index=iterations) \
                          .join(pd.DataFrame(ratios_array, index=iterations))

  return totals_and_ratios_df
```

```
def process_item_types(df, processing_col='item_type',␣
 ↪iterations=gta_account_ids):
  """
  Designed to return totals of each item type category per account_id.
  """
  concise_df = df[['account_id', processing_col]]
  array = []
  for id in iterations:
    id_df = find_subset_in_df('account_id', id, df)
    grouped = id_df.groupby(['account_id', processing_col]) \
                          .size() \
                          .reset_index()
    processed_dict = {item: size
                      for item, size
                      in zip(grouped[processing_col], grouped[0])}
    array.append(processed_dict)
  item_df = pd.DataFrame(array, index=iterations)
  return item_df


def find_subset_in_df(subset_col, desired_value, df):
  """
  Returns subset of `df` where `subset_col`
    is `desired_value`.
  """
  subset_df = df[df[subset_col] == desired_value]
  return subset_df


def calculate_listlike_ratio(series):
  """
  Sum series and divide it by length of itself.
  """
  ratio = np.sum(series) / len(series)
  return np.round(ratio, 2)


def process_ratios(df, cols_of_interest, iterations=gta_account_ids):
  """
  Find ratios of cols_of_interest,
    returns a df indexed by iterations (default is gta_account_ids).
  """
  ratios_array = []
  trimmed_df = df[cols_of_interest]

  for id in iterations:
```

```
    ## Only values in df referencing ID.
    id_df = find_subset_in_df('account_id', id, trimmed_df) #dict_df[id]

    ## Ratios
    ratios = {col+' ratio': calculate_listlike_ratio(id_df[col])
            for col
            in cols_of_interest}
    ratios_array.append(ratios)

  ratios_df = pd.DataFrame(ratios_array, index=iterations)
  return ratios_df


def process_sums(df, cols_of_interest, iterations=gta_account_ids):
  """
  Find ratios of cols_of_interest,
    returns new df of ratios, indexed by iterations
    (default is gta_account_ids).
  """
  totals_array = []
  trimmed_df = df[cols_of_interest]

  for id in iterations:
    ## Only values in df referencing ID.
    id_df = find_subset_in_df('account_id', id, trimmed_df) #dict_df[id]

    ## Totals
    totals = {col+' total': id_df[col].sum()
              for col
              in cols_of_interest}
    totals_array.append(totals)

  totals_df = pd.DataFrame(totals_array, index=iterations)
  return totals_df

def concatenate_into_series(*args):
  """
  Creates a concatenated series from unlimited *args.
  """
  concatenated_series = pd.concat(list(args))
  return concatenated_series


# Reference Lab 3
# The dataframe passed to VIF must include the intercept term. We add it the␣
  ↪same way we did before.
def VIF(df, columns):
```

```
    values = sm.add_constant(df[columns]).values
    num_columns = len(columns)+1
    vif = [variance_inflation_factor(values, i) for i in range(num_columns)]
    return pd.Series(vif[1:], index=columns)
```

2.2 Variables:

CSVs:

- gta_item_spend
- gta_player_activity
- gta_player_statistics
- gta_csvs_dict

Columns For Processing - activity_cols_of_interest - items_cols_of_interest - statistics_cols_of_interest

Processed CSV Data - processed_activity - DF containing summed and ratioed (player data / length that player was recorded) columns in activity_cols_of_interest - processed_items - DF containing counts of item categories purchased, for each player - processed_stats - DF containing summed columns from statistics_cols_of_interest

3 Data Processing

3.1 *Fill Incorrect values*

[ ]:
```
## The missing values in this specific dataset are only cars.
gta_item_spend['item_type'] = gta_item_spend['item_type'].fillna('CAR')
```

[ ]:
```
#Correct improper vehicle names
improper_names = {
    "Imponte Duke O Death": "Imponte Duke O'Death",
    "Lampadati Viseris": "Lampadati Viseris",
    "Albany Fr nken Stange": "Albany Fränken Stange",
    "P-45 Nokota": "P-45 Nokota"
}

gta_item_spend['item'] = gta_item_spend['item'].replace(improper_names)
```

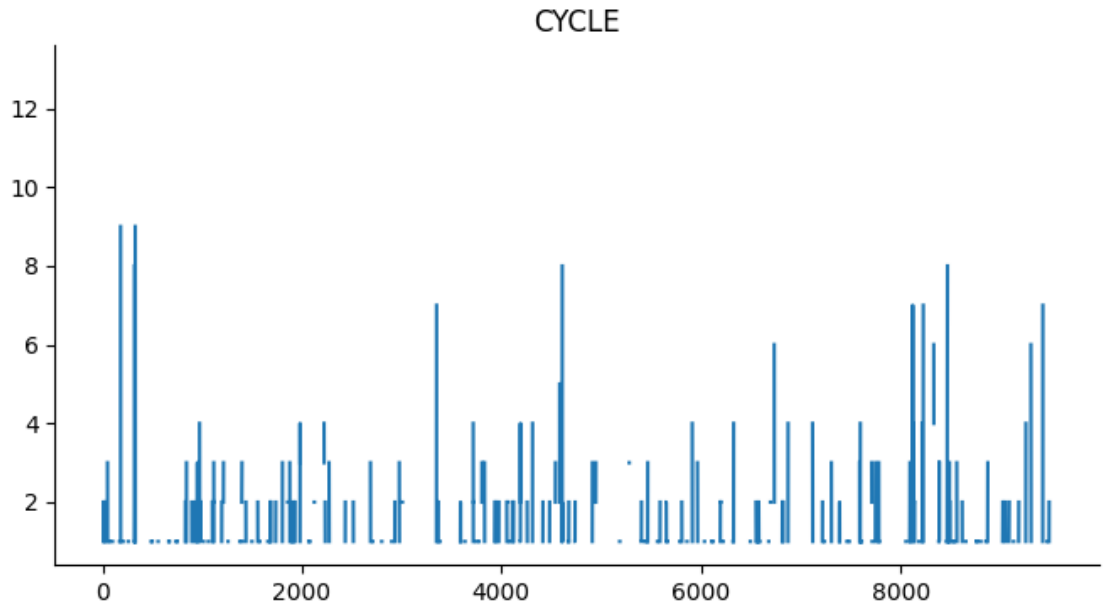
##Create DFs of Processed Interest Data

[ ]:
```
# Player Activity CSV
activity_cols_of_interest = ['account_id', 'kills', 'deaths', 'money_earned',
                             'rp_earned', 'success', 'money_vs_time_spent',
                             'rp_vs_time_spent', 'time_spent']

# Item Spend CSV
item_cols_of_interest = ['account_id', 'item_type']
```

```
# Player Statistics CSV
statistics_cols_of_interest = ['account_id', 'ltd_days_played', 'evc_balance',
                               'pvc_balance', 'char_rank', 'daily_playtime']
```

[ ]:
```
## Runtime is approx 1.5 minutes

processed_activity = process_sums_ratios(gta_player_activity,
                                 activity_cols_of_interest)
processed_items = process_item_types(gta_item_spend)
processed_stats = process_sums(gta_player_statistics,
                               statistics_cols_of_interest) \
                               .join(gta_player_statistics.groupby('account_id')
                                       ['char_rank'].max())
```

[ ]:
```
# @title CYCLE

from matplotlib import pyplot as plt
processed_items['CYCLE'].plot(kind='line', figsize=(8, 4), title='CYCLE')
plt.gca().spines[['top', 'right']].set_visible(False)
```



4 Modeling:

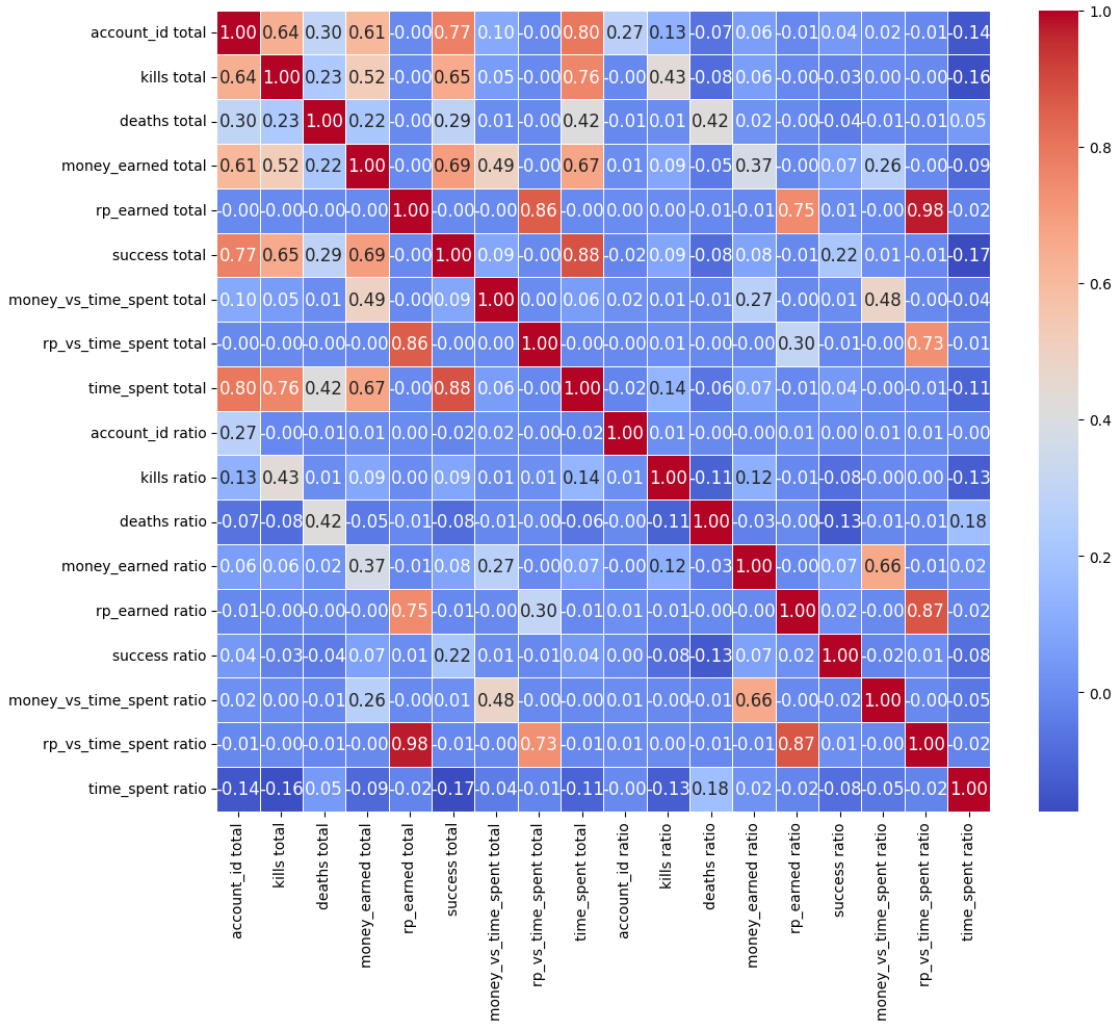
4.1 *Features*

4.1.1 *Feature Exploration*

[ ]:
```
# processed_activity.columns.tolist()
# processed_items.columns.tolist()
# processed_stats.columns.tolist()
```

[ ]:
```
corr = processed_activity.corr()

plt.figure(figsize=(12, 10))
sns.heatmap(corr, annot=True, fmt=".2f", annot_kws={"size": 12},␣
 ↪cmap='coolwarm', linewidths=.5)
```

[ ]: <Axes: >



[ ]:
```
processed_data = processed_activity[selected_features + [target]] \
                    .join(pd.DataFrame(processed_items.sum(axis='columns'))) \
                    .join(processed_stats)
processed_data['char_rank highest'] = gta_player_statistics.
 ↪groupby('account_id')['char_rank'].max()
```

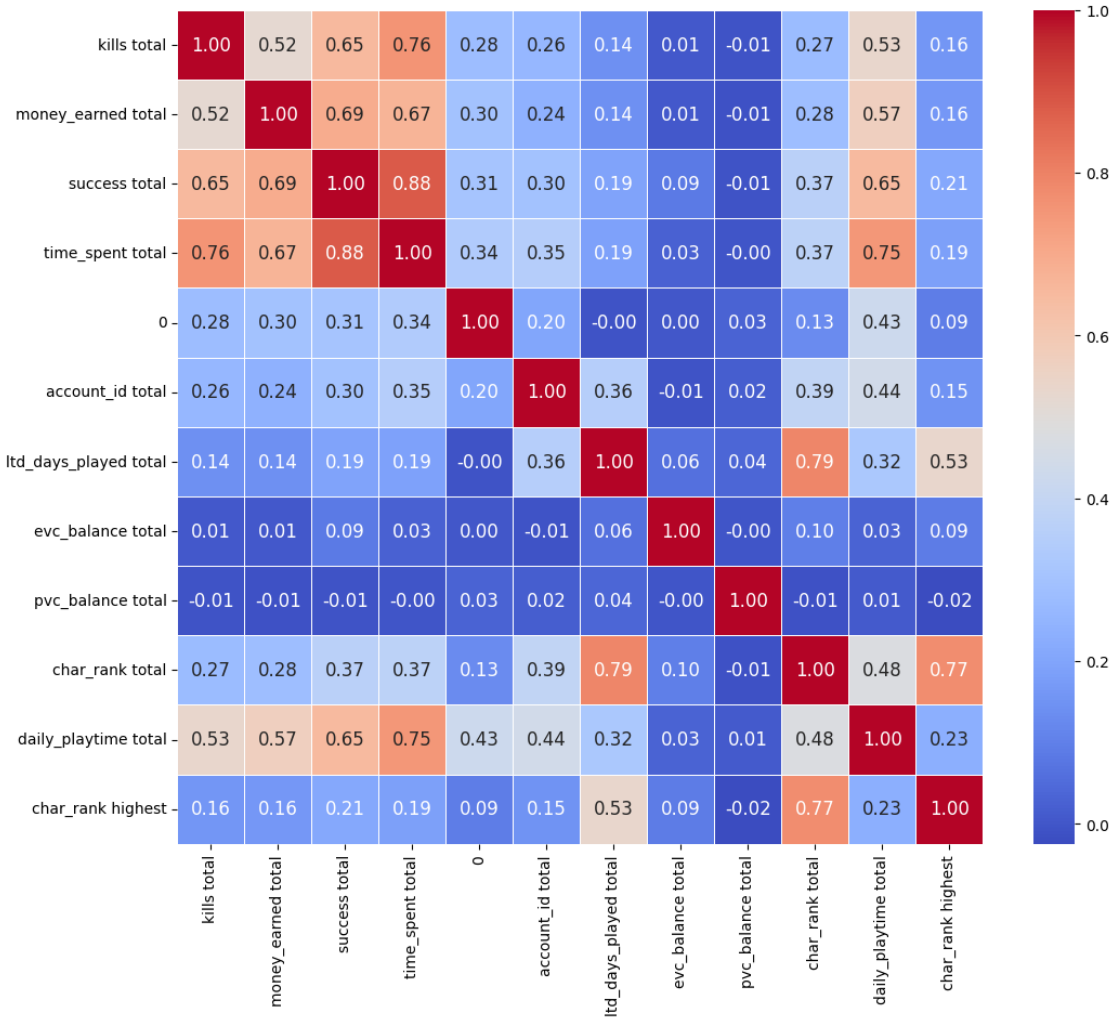
[ ]:
```
processed_data
```

[ ]:
```
      kills total  money_earned total  success total  time_spent total     0  \
0             236        4.424020e+04             14        134.700495  22.0
1              71        1.066346e+05             44        129.793234   4.0
2              33        6.118414e+05             29        159.093987  12.0
3              10        1.116525e+06             75        287.686893  20.0
4               0        5.196781e+03              0         17.958095  44.0
…             …                   …              …                …   …
9522            0        7.561316e+02              1          4.545949  10.0
9523            0        1.195260e+04             23         41.390627   3.0
9524          105        1.229039e+02              1         18.683499   3.0
9525            0        9.094366e+02              0          5.225844   1.0
9526           49        7.602630e+03              4         35.303681   1.0

      account_id total  ltd_days_played total  evc_balance total  \
0                    0                   6844       1.008279e+07
1                   39                   5206       9.559764e+05
2                   82                   3198       1.104105e+07
3                  102                    943       1.492467e+07
4                  104                  14761       1.678057e+06
…                    …                      …                  …
9522            190440                    795       2.352101e+07
9523             95230                    968       8.267912e+09
9524            133336                   3223       2.414774e+06
9525             95250                   9263       1.168961e+07
9526            152416                   1860       3.477978e+06

      pvc_balance total  char_rank total  daily_playtime total  \
0          163047.677191             1329             10.101785
1               0.000000             1534             13.870489
2               0.000000              762             17.796335
3          984721.370600              674             22.613138
4               0.000000             2566             14.678832
…                     …                …                     …
9522            0.000000              197              2.465090
9523            0.000000              549              4.248382
9524            0.000000              574              6.578154
9525            0.000000             1140              0.578406
```

```
9526           0.000000              484              9.518471

      char_rank highest
0                    36
1                    44
2                    23
3                    28
4                   100
…                   …
9522                 10
9523                 56
9524                 41
9525                114
9526                 32

[8772 rows x 12 columns]
```

[ ]:
```
all_corr = processed_data.corr()

plt.figure(figsize=(12, 10))
sns.heatmap(all_corr, annot=True, fmt=".2f", annot_kws={"size": 12},␣
 ↪cmap='coolwarm', linewidths=.5)
```

[ ]:
```
<Axes: >
```



4.1.2 *Feature Selection*

[ ]:
```
target = 'time_spent total'
selected_features = ['kills total', 'money_earned total', 'success total']

X = processed_activity[selected_features]
y = processed_activity[target]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

4.2 *Baseline Model* – 25-30% accuracy

Predict a random value within one standard deviation of the mean and median of the actual value.

Average accuracy is ~25%.

[ ]:
```
#mean/median and standard deviation of variable
mean, median, std = y.mean(), \
                    y.median(), \
                    y.std()

#upper and lower limits
upper_mean, upper_median = (mean + std), \
                           (median + std)
lower_mean, lower_median = max(mean-std, 0), \
                           max(median-std, 0)


#performance of the baseline
baseline_pred_mean = np.where(y > mean, \
                              random.choice(np.arange(mean, upper_mean)),
                              random.choice(np.arange(lower_mean, mean)))

baseline_pred_median = np.where(y > mean,
                                random.choice(np.arange(median,␣
 ↪upper_median)),
                                random.choice(np.arange(lower_median,␣
 ↪median)))


# MSE, MAE, R2 Score
mse_bl_mean = mean_squared_error(y, baseline_pred_mean)
mae_bl_mean = mean_absolute_error(y, baseline_pred_mean)
r2_bl_mean = r2_score(y, baseline_pred_mean)

mse_bl_median = mean_squared_error(y, baseline_pred_median)
mae_bl_median = mean_absolute_error(y, baseline_pred_median)
r2_bl_median = r2_score(y, baseline_pred_median)

print(f"MSE (Mean): {mse_bl_mean}")
print(f"MAE (Mean): {mse_bl_mean}")
print(f"R2 Score (Mean): {r2_bl_mean}")
print(f"Baseline Model Accuracy (Mean): {np.round(r2_bl_mean * 100, 2)}% \n")

print(f"MSE (Median): {mse_bl_median}")
print(f"MAE (Median): {mse_bl_median}")
print(f"R2 Score (Median): {r2_bl_median}")
print(f"Baseline Model Accuracy (Median): {np.round(r2_bl_median * 100, 2)}%")
```

```
MSE (Mean): 83408.13237131125
MAE (Mean): 83408.13237131125
R2 Score (Mean): 0.29135713423644183
Baseline Model Accuracy (Mean): 29.14%
```

```
MSE (Median): 109420.87037988885
MAE (Median): 109420.87037988885
R2 Score (Median): 0.07035061263381381
Baseline Model Accuracy (Median): 7.04%
```

4.3 *Random Forest Regression* – 82% accuracy

[ ]:
```
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
forest_regressor_r2 = r2_score(y_test, y_pred)

y_pred, mse, forest_regressor_r2

print(f'Forest Regressor Model accuracy is: {np.round(forest_regressor_r2, 2) *␣
↪100}%')
```

```
Forest Regressor Model accuracy is: 82.0%
```

4.4 *Lasso Regression* – 84% accuracy

[ ]:
```
lasso_reg = linear_model.Lasso(alpha=50, max_iter=100, tol=0.1)

lasso_reg.fit(X_train, y_train)
lasso_reg.score(X_test, y_test)
lasso_test_prediction_score = lasso_reg.score(X_test, y_test)

print(f'LASSO Model accuracy is: {np.round(lasso_test_prediction_score, 2) *␣
↪100}%')
```

```
LASSO Model accuracy is: 84.0%
```

4.5 *Ridge Regression* – 83% accuracy

[ ]:
```
ridge_regression = Ridge(alpha = 50, max_iter = 100, tol = 0.1)


ridge_regression.fit(X_train, y_train)
ridge_regression.score(X_test, y_test)
ridge_test_prediction_score = ridge_regression.score(X_test, y_test)

print(f'RIDGE Model accuracy is: {np.round(ridge_test_prediction_score, 2) *␣
↪100}%')
```

```
RIDGE Model accuracy is: 83.0%
```