Econ143Final

May 9, 2024

0.0.1 In Clash Royale's triple draft mode, what spell cards are the best choices?

Econ 143 Final Project: By Lawrence Chen

[ ]:
```
# Setup Cell

import pandas as pd

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import random
import statsmodels.api as sm
import time
import ast
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

pd.set_option('display.max_columns', 500)
pd.set_option('display.max_rows', 100)

np.set_printoptions(threshold=np.inf)
```

** Note to Professor/Grader: Please feel free to skip or glimpse over the following background of Clash Royale the game, as the context isn't necessary to understand my analysis. The bulk of my work is provided after the cell labeled "Analysis."

Sources to data: provided in the following jupyter notebook to this: Creating the Data

0.0.2 Background to Clash Royale

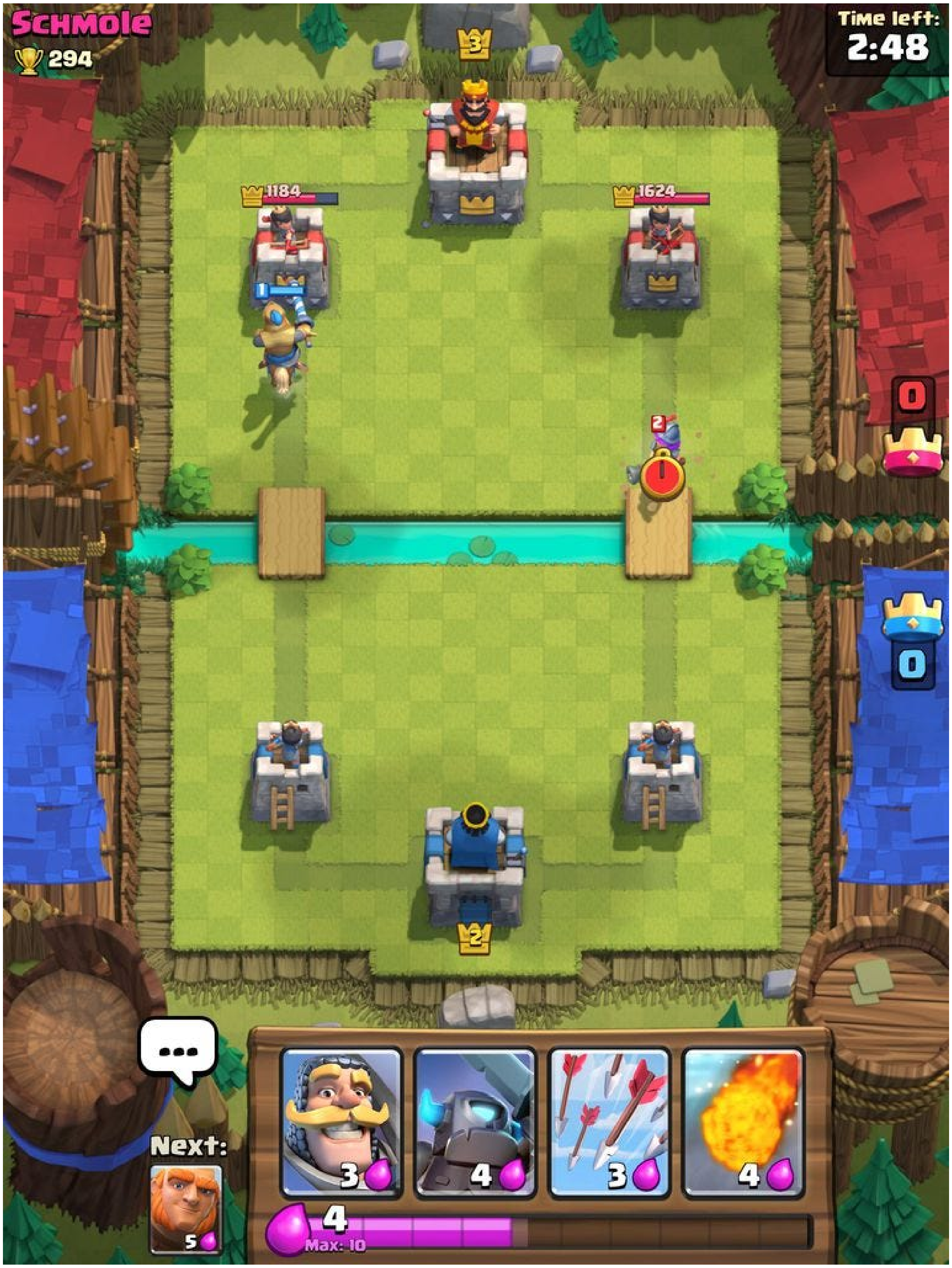
Clash Royale is a popular strategic mobile game that features a battle between two players. In brief, each player's objective is to take down as many of their opponent's 3 towers in a 3-6 minute span, hence winning the game. Players play in real-time, which means players may play moves simulatenously. In other words, there is no set order to moves, unlike a turn-based game like Chess.

To gain a strategic advantage over their opponent, a player chooses a deck that consists of 8 cards. Cards in Clash Royale are for the most part, not directly comparable to each other, as there are several features that differentiate cards from one another. This is similar to how a banana is priced lower than a MacBook, but provides more nutritional benefits.

An example of gameplay:

```
[ ]: from IPython.display import Image
image_path = 'gameplay_1.jpeg'
Image(filename=image_path)
```

[ ]:



A big issue in Clash Royale is the existence of inequalities that players face before starting a match. For highly-skilled Clash Royale players, certain decks will offer disproportionately significant advantages over others, in comparison to the effects of being at a higher skill level. In addition, this relationship is not transitive. This means that while most of the time deck A > deck B and deck B > deck C, it isn't always the case that deck A > deck C. As a result, a game's outcome is severely affected on the deck combinations a player chooses.

In summary, winning in Clash Royale can be largely attributed to chance, which hugely diminishes the impact of skill level on winning. (If I can beat Roger Federer in a tennis match 25% of the time, who's to say I can't win Wimbledon?) Fortunately, Clash Royale offers various modes that are designed to ensure equality between equally-skilled players, one of which I have chosen to analyze deeper below.

0.0.3 A Brief Background into Triple Draft Mode:

In Clash Royale's Triple Draft Mode, each of the two players are faced with 8 choices. They are presented with 3 cards and select 1 card to add to their deck, while trashing the other cards. In total, they choose 4 cards by themselves, and are forced to take 4 cards as given by their opponent. It should be noted that under this mode, only 16 cards are actually available from the players to pick from.

In this manner, Triple Draft Mode provides an excellent microcosm to analyze Clash Royale under. First, we see that all the cards (even the bad ones) will appear for selection, as they are randomly selected. This removes any survivorship bias that may have existed, as the distribution of cards should be uniformly distributed. Second, players have a limited amount of options for creating their deck. This means that the aforementioned advantage that a deck has over another is largely a function of better choice making by 1 player. Third, the amount of synergy cards have is limited, due to there only being 24 possible draft choices among both players. It becomes easier to isolate the impact that individual cards have on winning a match.

As a result, the winning player will probably be the one who makes better choices in creating their deck and plays a better match. Draft Mode provides a fair(er) playing field for a competitive match and thus a good starting point for analyzing Clash Royale.

An example of a draft choice between spells:

[ ]:
```
image_path = 'tripdraft.jpeg'
Image(filename=image_path)
```

[ ]:


Opponent Chooses From
3
4
4
Choose Your Card
6
6
4
Battle Begins In:
48sec

0.1 Analysis

Main Research Question: What spells are better choices than others? Can we quantify the impact of choosing one over the other? To win a match in Clash Royale, it's important to building the best possible deck before entering a match. By determining which cards are the best, we can therefore increase our chances of winning.

However, as previously mentioned, cards are often not directly comparable, much like apples and oranges. So, I've decided to limit the analysis to comparing a type of card (Spells).

There are currently 14 spell cards that do direct damage (excluding Void as it was recently added): Arrows, Zap, Giant Snowball, Royal Delivery, Fireball, Rocket, Earthquake, Lightning, Freeze, Barbarian Barrel, Poison, Rage, Tornado, and The Log. All of these spell cards can be reasonably interchanged with each other in draft mode in various contexts.

[ ]:
```
image_path = 'tierlist.png'
Image(filename=image_path)
```

[ ]:

- The importance of spells in Clash Royale is very crucial, so much so that players are in constant debate over which ones are the best. Source: Spells Tier List - Reddit

0.1.1 Data

The dataset I used includes the results of 306246 matches played: * column "presult": 0 if the match was won, 1 if the match was lost. (Note: There are equal numbers of 0s and 1s in this dataset.) * ptag: the player who played this match

The next 14 columns are boolean values that gather whether a spell was played in a match or not: 0 if no, 1 if yes.

- expLevel: A player's account level: A good indicator of how long a player has been playing the game, or how much experience they have.

- trophies: The number of trophies a player has: A good indicator of the skill of a player. For instance, a comparatively low expLevel but high trophy level indicates that the player is really good, while a comparatively high expLevel but low trophy level indicates that the player is bad.
- num_spells_used: How many of the 14 spells did a player use? Calculated by summming up the boolean columns.

[ ]:
```
df = pd.read_csv("triple_draft_data.csv")
df["num_spells_used"] = df.drop(df.columns[[0, 1, 2, -2, -1]], axis = 1).
 ↪sum(axis=1)
print(df.shape)
df.head()
# 153123 unique matches: 2 rows for each match (306246 total) since each match␣
 ↪has a winner and a loser
```

```
(306246, 20)
```

[ ]:
```
   Unnamed: 0  presult       ptag  fireball_bool  arrows_bool  rage_bool  \
0           0        0   PRQJRGJ8              0            0          0
1           1        1  P2C2CCQ2V              0            0          0
2           2        1  9JVCJ92V9              1            0          0
3           3        1  Y82RVYPC8              0            1          1
4           4        1   Y89UL2JL              0            0          0

   rocket_bool  freeze_bool  lightning_bool  zap_bool  poison_bool  \
0            0            0               0         0            0
1            0            0               0         0            0
2            0            0               0         0            0
3            0            0               0         0            1
4            0            0               0         0            1

   the_log_bool  tornado_bool  earthquake_bool  barbarian_barrel_bool  \
0             0             0                0                      0
1             0             0                0                      1
2             1             0                0                      0
3             0             0                0                      0
4             0             0                0                      0

   snowball_bool  royal_delivery_bool  expLevel  trophies  num_spells_used
0              0                    0        13      5518                0
1              0                    0        13      6487                1
2              0                    0        13      6600                2
```

```
3                0                     0        13      5695                    3
4                0                     0        13      5885                    1
```

Data Exploration

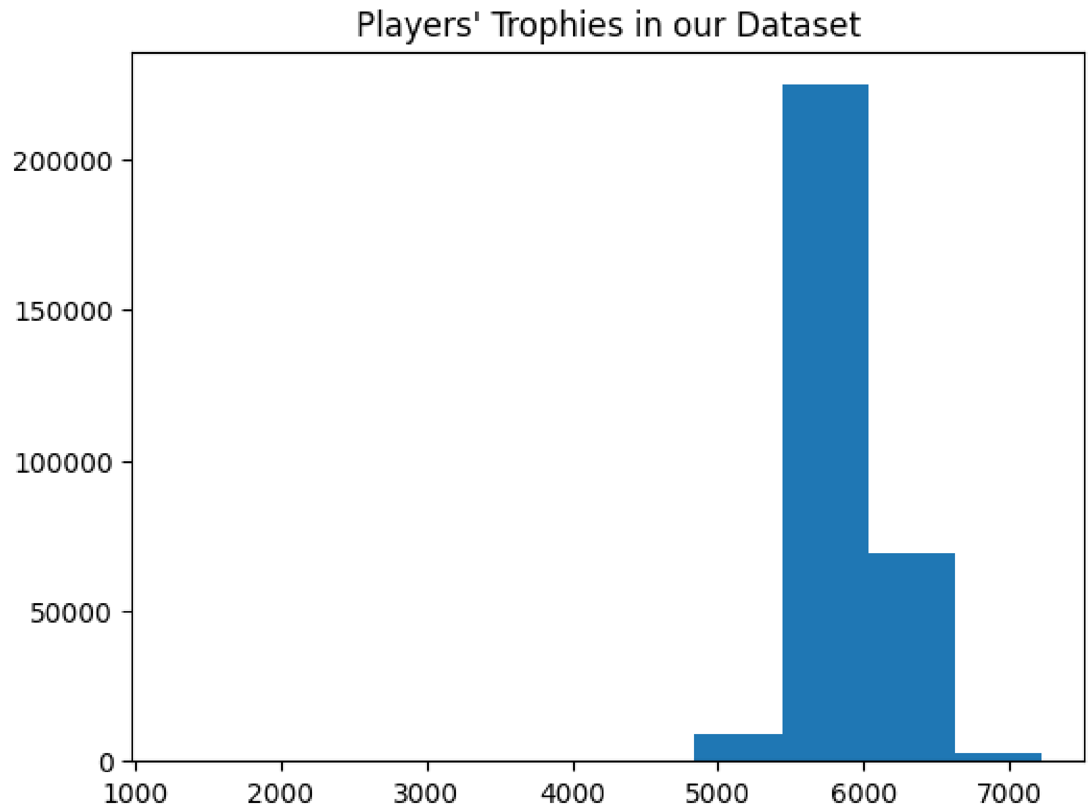
```
[ ]: # Summary Statistics for the data



     '''
     Interpretations:
     * The mean displays the usage rate of the 14 spells. We see that the arrows are␣
      ↪most commonly used in matches at 22.64%, while the freeze is used the least␣
      ↪at 2%.
     * We see from the percentiles that a majority of players use above 2 spells in␣
      ↪a deck. One player even used 6!
     * The percentiles also show that our data is mostly comprised of high level␣
      ↪players: as most are level 13 and have a high trophy count.
     '''

     df.describe()
```

```
[ ]:          Unnamed: 0        presult   fireball_bool    arrows_bool  \
     count  306246.000000  306246.000000  306246.000000  306246.000000
     mean   153122.500000       0.537117       0.209322       0.226419
     std     88405.749607       0.498621       0.406825       0.418514
     min         0.000000       0.000000       0.000000       0.000000
     25%     76561.250000       0.000000       0.000000       0.000000
     50%    153122.500000       1.000000       0.000000       0.000000
     75%    229683.750000       1.000000       0.000000       0.000000
     max    306245.000000       1.000000       1.000000       1.000000

                rage_bool    rocket_bool    freeze_bool  lightning_bool  \
     count  306246.000000  306246.000000  306246.000000   306246.000000
     mean        0.027289       0.053741       0.022743        0.191741
     std         0.162923       0.225506       0.149084        0.393671
     min         0.000000       0.000000       0.000000        0.000000
     25%         0.000000       0.000000       0.000000        0.000000
     50%         0.000000       0.000000       0.000000        0.000000
     75%         0.000000       0.000000       0.000000        0.000000
     max         1.000000       1.000000       1.000000        1.000000

                 zap_bool    poison_bool   the_log_bool   tornado_bool  \
     count  306246.000000  306246.000000  306246.000000  306246.000000
     mean        0.139845       0.214703       0.206830       0.070956
     std         0.346827       0.410617       0.405034       0.256752
     min         0.000000       0.000000       0.000000       0.000000
     25%         0.000000       0.000000       0.000000       0.000000
```

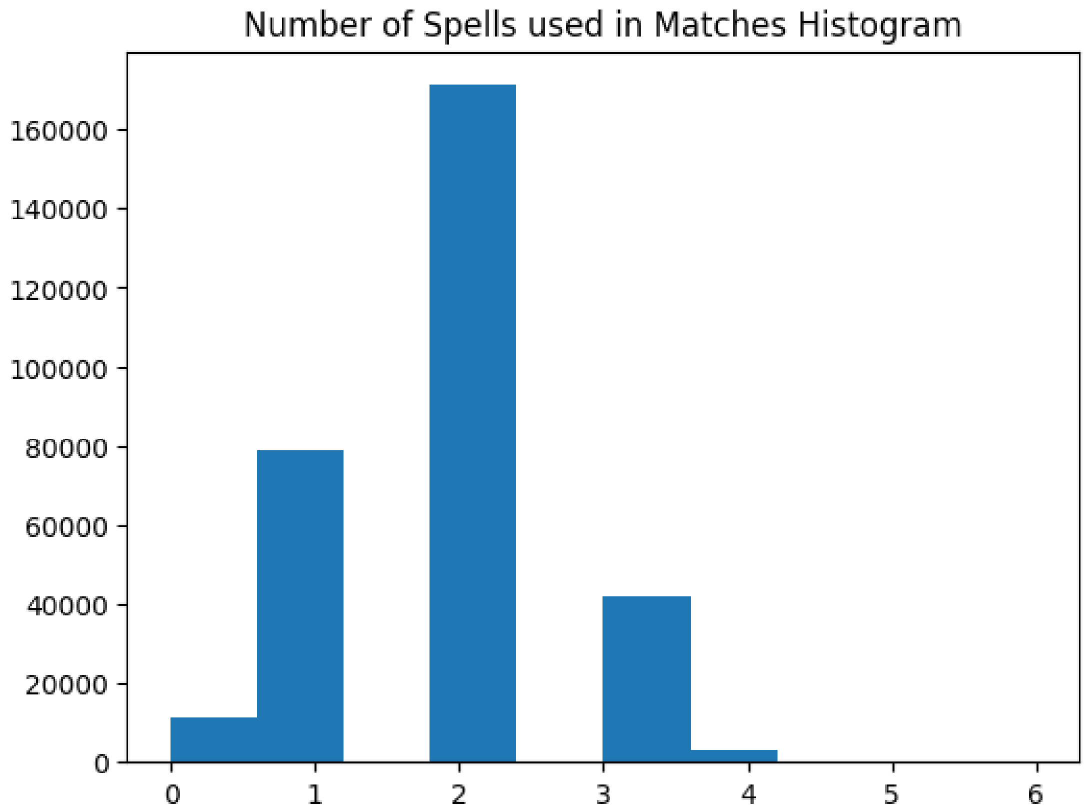
```
50%          0.000000        0.000000        0.000000        0.000000
75%          0.000000        0.000000        0.000000        0.000000
max          1.000000        1.000000        1.000000        1.000000

       earthquake_bool  barbarian_barrel_bool  snowball_bool  \
count    306246.000000          306246.000000  306246.000000
mean          0.125249               0.153289       0.103058
std           0.331002               0.360266       0.304035
min           0.000000               0.000000       0.000000
25%           0.000000               0.000000       0.000000
50%           0.000000               0.000000       0.000000
75%           0.000000               0.000000       0.000000
max           1.000000               1.000000       1.000000

       royal_delivery_bool       expLevel       trophies  num_spells_used
count        306246.000000  306246.000000  306246.000000    306246.000000
mean              0.078927      12.928531    5915.364981         1.824112
std               0.269625       0.315888     284.954935         0.743603
min               0.000000       6.000000    1275.000000         0.000000
25%               0.000000      13.000000    5749.000000         1.000000
50%               0.000000      13.000000    5877.000000         2.000000
75%               0.000000      13.000000    6023.000000         2.000000
max               1.000000      13.000000    7221.000000         6.000000
```

[ ]:

```
# Skill level of players
plt.hist(df["trophies"])
plt.title("Players' Trophies in our Dataset") ;

# Our dataset includes highly skilled players - good, since they care more for
 ↪optimal play than an inexperienced or casual player.
```

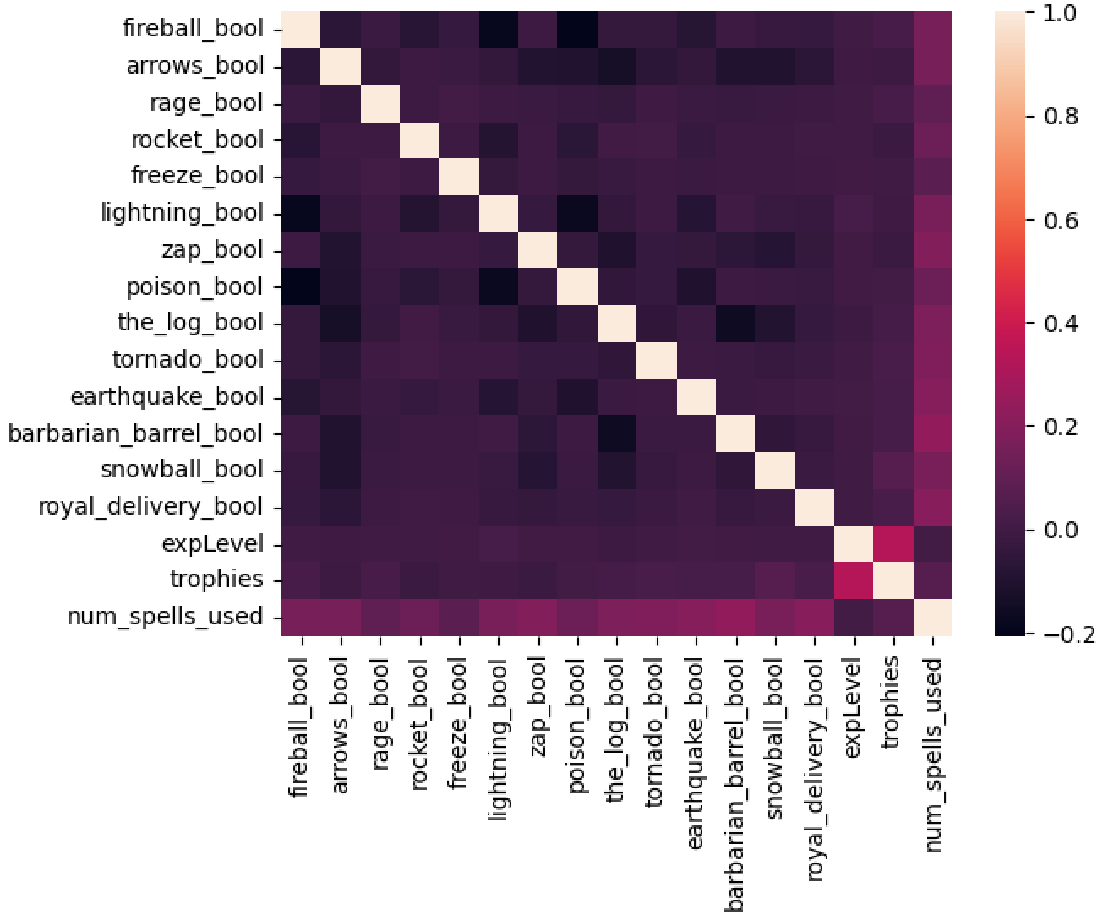


[ ]:
```
plt.hist(df["num_spells_used"])
plt.title("Number of Spells used in Matches Histogram");

# 2 is the mode -> Most players like to select 2 spells
```



[ ]:
```
correlation_matrix = df.drop(df.columns[[0, 1, 2]], axis = 1).corr()
sns.heatmap(correlation_matrix)

# Close to no correlation between any variables: As desired, choices of cards␣
↪are largely independent.
# This assumption is important when interpreting the results, as we focus on␣
↪the impact of individual choices.
# num_spells is a sum of the boolean variables (which is by default related.)
```

[ ]: <Axes: >



[ ]:
```
df.drop(df.columns[[0, 1, 2]], axis = 1).sum(axis=0)
# Marginals of the columns: Arrows, Poisson, Fireball, and the Log are the most␣
↪commonly picked cards among this sample of games.
```

[ ]:
```
fireball_bool            64104
arrows_bool              69340
rage_bool                 8357
rocket_bool              16458
freeze_bool               6965
lightning_bool           58720
zap_bool                 42827
poison_bool              65752
the_log_bool             63341
tornado_bool             21730
earthquake_bool          38357
barbarian_barrel_bool    46944
snowball_bool            31561
royal_delivery_bool      24171
```

```
expLevel                      3959311
trophies                   1811556864
num_spells_used                558627
dtype: int64
```

Understanding the Data: * Arrows, Poisson, Fireball, and the Log being the most commonly picked cards in this game mode isn't surprising. All 4 choices are very robust: choosing these cards are often regarded as a safe choice, which will not automatically lose or win you the game.

- The mode of the spells chosen histogram is 2. Players like to reserve 2 out of their 8 deck slots for spells in order to build a flexible deck.

0.2 Methodology: Logistic Regression

My goal with this data was to determine which spell choices contribute most to winning in Clash Royale. To do so, I did a logistic regression.

Independent Variable: - "presult" column with 0s and 1s - whether the player won or not

Dependent Variables: - All other numerical variables in the dataset, except for fireball_bool: to ensure linear independence between variables.

[ ]:

```python
# Logistic Regression

X = df.drop(df.columns[[0, 1, 2, 3]], axis = 1)  # Dropping the first 4 columns
X = sm.add_constant(X)     # Adding a constant: Origin is unrealistic
y = df["presult"]          # Dependent Variabel: Outcome of the Match

model = sm.Logit(y, X)     # Logistic Regression Model trained on y, X
result = model.fit()       # Fitting the model

print(result.summary())    # Outputs regression table of model results
```

```
Optimization terminated successfully.
         Current function value: 0.688254
         Iterations 4
                           Logit Regression Results
==============================================================================
Dep. Variable:                presult   No. Observations:               306246
Model:                          Logit   Df Residuals:                   306229
Method:                           MLE   Df Model:                           16
Date:                Thu, 09 May 2024   Pseudo R-squ.:                0.003093
Time:                        16:42:10   Log-Likelihood:            -2.1078e+05
converged:                       True   LL-Null:                   -2.1143e+05
Covariance Type:            nonrobust   LLR p-value:                1.100e-268
================================================================================
=========
                           coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
```

```
---------
const                    -1.3392      0.152     -8.832      0.000      -1.636
-1.042
arrows_bool               0.0986      0.012      8.007      0.000       0.074
0.123
rage_bool                -0.0713      0.024     -2.982      0.003      -0.118
-0.024
rocket_bool              -0.2155      0.018    -12.063      0.000      -0.251
-0.180
freeze_bool              -0.1326      0.026     -5.173      0.000      -0.183
-0.082
lightning_bool           -0.0562      0.012     -4.762      0.000      -0.079
-0.033
zap_bool                 -0.0508      0.014     -3.640      0.000      -0.078
-0.023
poison_bool               0.0450      0.011      3.925      0.000       0.023
0.067
the_log_bool              0.1532      0.013     11.975      0.000       0.128
0.178
tornado_bool             -0.0839      0.017     -5.041      0.000      -0.117
-0.051
earthquake_bool          -0.1212      0.014     -8.884      0.000      -0.148
-0.094
barbarian_barrel_bool     0.1009      0.014      7.324      0.000       0.074
0.128
snowball_bool            -0.0277      0.015     -1.847      0.065      -0.057
0.002
royal_delivery_bool      -0.0459      0.016     -2.849      0.004      -0.077
-0.014
expLevel                  0.0089      0.012      0.729      0.466      -0.015
0.033
trophies                  0.0002   1.36e-05     17.014      0.000       0.000
0.000
num_spells_used          -0.0076      0.010     -0.774      0.439      -0.027
0.012
==============================================================================
=========
```

Overall Interpretation & Implications for Strategy: This regression table provides useful strategic insights for players who want to win more Clash Royale Games:

We see that a couple of spells have positive relationships with winning: arrows, poison, the log, and barbarian barrel. This corresponds nicely with the spells that are most commonly selected by players, which shows that players make good decisions when choosing spells to pick.

There are also positive relationships with expLevel and trophies: higher skilled players will win more games. In addition, there is a negative relationship with the number of spells to choose. Spells are good, but lmiiting the number to pick is

We can interpret the coefficients as saying: If I pick this spell and no others, I will have boosted my chances of winning by a proportion of {coeff}. For example, picking arrows corresponds to a coefficient of 0.0986, which means an increase of just under 10% of winning.

0.3 Methodology 2: Logistic Regression + Quantiles

- Controlling for Trophy Quantiles
- It may be the case that better players (higher trophy counts) prefer different cards.

```
[ ]: # Quantiles
print(df["trophies"].quantile(0.25))
print(df["trophies"].quantile(0.50))
print(df["trophies"].quantile(0.75))
print(df["trophies"].quantile(1.00))
```

```
5749.0
5877.0
6023.0
7221.0
```

Hence, I repeated the above process for different quantile groups of players.

Quantile 1: 0 to 25th percentile:

```
[ ]: df_1 = df[df["trophies"] <= 5749]

X = df_1.drop(df_1.columns[[0, 1, 2, 3]], axis = 1)
X = sm.add_constant(X)
y = df_1["presult"]

model = sm.Logit(y, X)
result = model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.689943
         Iterations 4
                           Logit Regression Results
==============================================================================
Dep. Variable:                presult   No. Observations:                76727
Model:                          Logit   Df Residuals:                    76710
Method:                           MLE   Df Model:                           16
Date:                Thu, 09 May 2024   Pseudo R-squ.:                0.003603
Time:                        16:42:11   Log-Likelihood:                -52937.
converged:                       True   LL-Null:                       -53129.
Covariance Type:            nonrobust   LLR p-value:                 1.442e-71
================================================================================
=========
                            coef    std err          z      P>|z|      [0.025
0.975]
```

```
------------------------------------------------------------------------------------
---------
const                  -2.3774      0.257     -9.265      0.000      -2.880
-1.874
arrows_bool             0.1211      0.025      4.930      0.000       0.073
0.169
rage_bool              -0.1051      0.050     -2.096      0.036      -0.203
-0.007
rocket_bool            -0.2037      0.035     -5.894      0.000      -0.271
-0.136
freeze_bool            -0.1660      0.051     -3.251      0.001      -0.266
-0.066
lightning_bool          0.0051      0.024      0.212      0.832      -0.042
0.052
zap_bool                0.0218      0.028      0.790      0.430      -0.032
0.076
poison_bool             0.0451      0.023      1.924      0.054      -0.001
0.091
the_log_bool            0.2141      0.026      8.381      0.000       0.164
0.264
tornado_bool           -0.0627      0.035     -1.783      0.075      -0.132
0.006
earthquake_bool        -0.1086      0.028     -3.853      0.000      -0.164
-0.053
barbarian_barrel_bool   0.1369      0.028      4.891      0.000       0.082
0.192
snowball_bool           0.0251      0.032      0.788      0.431      -0.037
0.087
royal_delivery_bool    -0.0528      0.034     -1.560      0.119      -0.119
0.014
expLevel               -0.0025      0.016     -0.154      0.878      -0.034
0.029
trophies                0.0004   5.43e-05      8.063      0.000       0.000
0.001
num_spells_used        -0.0220      0.019     -1.137      0.256      -0.060
0.016
====================================================================================
=========
```

Differences and Findings Among the lower quantile of players, more cards have a positive relationship with winning, such as the lightning. This means that more cards are viable at lower levels.

Quantile 2: 25th to 50th percentile:

[ ]:
```
df_2 = df[(df["trophies"] <= 5877) & (df["trophies"] > 5749)]

X = df_2.drop(df_2.columns[[0, 1, 2, 3]], axis = 1)
```

```python
X = sm.add_constant(X)
y = df_2["presult"]

model = sm.Logit(y, X)
result = model.fit()

print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.689292
         Iterations 4
                           Logit Regression Results
==============================================================================
Dep. Variable:                presult   No. Observations:                76732
Model:                          Logit   Df Residuals:                    76715
Method:                           MLE   Df Model:                           16
Date:                Thu, 09 May 2024   Pseudo R-squ.:                0.002907
Time:                        16:42:12   Log-Likelihood:                -52891.
converged:                       True   LL-Null:                       -53045.
Covariance Type:            nonrobust   LLR p-value:                 4.632e-56
================================================================================
=========
                          coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
---------
const                   0.0747      1.278      0.058      0.953      -2.430
2.580
arrows_bool             0.1304      0.025      5.271      0.000       0.082
0.179
rage_bool              -0.0301      0.050     -0.602      0.547      -0.128
0.068
rocket_bool            -0.2353      0.035     -6.661      0.000      -0.305
-0.166
freeze_bool            -0.1450      0.050     -2.880      0.004      -0.244
-0.046
lightning_bool         -0.0813      0.023     -3.467      0.001      -0.127
-0.035
zap_bool               -0.0322      0.028     -1.164      0.244      -0.086
0.022
poison_bool             0.0545      0.023      2.355      0.019       0.009
0.100
the_log_bool            0.1681      0.026      6.512      0.000       0.117
0.219
tornado_bool           -0.0683      0.034     -2.032      0.042      -0.134
-0.002
earthquake_bool        -0.1139      0.027     -4.160      0.000      -0.168
-0.060
```

```
barbarian_barrel_bool     0.1236      0.028      4.486      0.000       0.070
0.178
snowball_bool            -0.0563      0.031     -1.794      0.073      -0.118
0.005
royal_delivery_bool      -0.0552      0.032     -1.700      0.089      -0.119
0.008
expLevel                 -0.0649      0.040     -1.629      0.103      -0.143
0.013
trophies                  0.0002      0.000      0.740      0.459      -0.000
0.001
num_spells_used          -0.0178      0.020     -0.908      0.364      -0.056
0.021
================================================================================
=========
```

Differences and Findings The findings at this quartile range are largely similar to that of the overall findings, except that there is a positive constant in this case. This means that the use of fireball, our omitted variable for linearity, is even more positive.

Quantile 3: 50th to 75th percentile:

[ ]:
```python
df_3 = df[(df["trophies"] <= 6023) & (df["trophies"] > 5877)]

X = df_3.drop(df_3.columns[[0, 1, 2, 3]], axis = 1)
X = sm.add_constant(X)
y = df_3["presult"]

model = sm.Logit(y, X)
result = model.fit()

print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.688612
         Iterations 4
                           Logit Regression Results
==============================================================================
Dep. Variable:                presult   No. Observations:                76323
Model:                          Logit   Df Residuals:                    76306
Method:                           MLE   Df Model:                           16
Date:                Thu, 09 May 2024   Pseudo R-squ.:                0.001773
Time:                        16:42:12   Log-Likelihood:                -52557.
converged:                       True   LL-Null:                       -52650.
Covariance Type:            nonrobust   LLR p-value:                 3.872e-31
================================================================================
=========
                            coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
```

```
---------
const                    0.0256      1.193      0.021      0.983      -2.312
2.364
arrows_bool              0.0880      0.025      3.566      0.000       0.040
0.136
rage_bool               -0.0318      0.049     -0.654      0.513      -0.127
0.064
rocket_bool             -0.1608      0.036     -4.433      0.000      -0.232
-0.090
freeze_bool             -0.0711      0.052     -1.367      0.172      -0.173
0.031
lightning_bool          -0.0444      0.023     -1.897      0.058      -0.090
0.001
zap_bool                -0.0695      0.028     -2.490      0.013      -0.124
-0.015
poison_bool              0.0424      0.023      1.862      0.063      -0.002
0.087
the_log_bool             0.1244      0.026      4.837      0.000       0.074
0.175
tornado_bool            -0.0583      0.033     -1.762      0.078      -0.123
0.007
earthquake_bool         -0.1039      0.027     -3.855      0.000      -0.157
-0.051
barbarian_barrel_bool    0.0832      0.027      3.053      0.002       0.030
0.137
snowball_bool           -0.0466      0.030     -1.565      0.117      -0.105
0.012
royal_delivery_bool     -0.0153      0.032     -0.480      0.632      -0.078
0.047
expLevel                -0.1328      0.050     -2.656      0.008      -0.231
-0.035
trophies                 0.0003      0.000      1.879      0.060   -1.36e-05
0.001
num_spells_used         -0.0223      0.020     -1.118      0.263      -0.061
0.017
================================================================================
=========
```

Differences and Findings Similar findings to the 25th to 50th percentile range. There are noticeably lower coefficients for most of our variables, which may reveal that choices to winning don't matter as much at this level.

Quantile 4: 75th to 100th percentile:

```
[ ]: df_4 = df[(df["trophies"] <= 7221) & (df["trophies"] > 6023)]

     X = df_4.drop(df_4.columns[[0, 1, 2, 3]], axis = 1)
     X = sm.add_constant(X)
```

```python
y = df_4["presult"]

model = sm.Logit(y, X)
result = model.fit()

print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.684395
         Iterations 4
                           Logit Regression Results
==============================================================================
Dep. Variable:                presult   No. Observations:                76464
Model:                          Logit   Df Residuals:                    76447
Method:                           MLE   Df Model:                           16
Date:                Thu, 09 May 2024   Pseudo R-squ.:                0.002691
Time:                        16:42:13   Log-Likelihood:                -52332.
converged:                       True   LL-Null:                       -52473.
Covariance Type:            nonrobust   LLR p-value:                 1.133e-50
================================================================================
=========
                           coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
---------
const                    1.5579      0.868      1.795      0.073      -0.144
3.259
arrows_bool              0.0531      0.025      2.152      0.031       0.005
0.101
rage_bool               -0.1132      0.044     -2.583      0.010      -0.199
-0.027
rocket_bool             -0.2567      0.037     -6.917      0.000      -0.329
-0.184
freeze_bool             -0.1381      0.052     -2.661      0.008      -0.240
-0.036
lightning_bool          -0.1020      0.023     -4.341      0.000      -0.148
-0.056
zap_bool                -0.1319      0.029     -4.621      0.000      -0.188
-0.076
poison_bool              0.0388      0.022      1.730      0.084      -0.005
0.083
the_log_bool             0.1062      0.025      4.179      0.000       0.056
0.156
tornado_bool            -0.1420      0.032     -4.475      0.000      -0.204
-0.080
earthquake_bool         -0.1587      0.027     -5.935      0.000      -0.211
-0.106
barbarian_barrel_bool    0.0598      0.028      2.167      0.030       0.006
```

```
0.114
snowball_bool          -0.0422      0.028     -1.506      0.132      -0.097
0.013
royal_delivery_bool    -0.0697      0.031     -2.243      0.025      -0.131
-0.009
expLevel               -0.1475      0.065     -2.278      0.023      -0.275
-0.021
trophies              9.34e-05   3.35e-05      2.791      0.005    2.78e-05
0.000
num_spells_used         0.0286      0.020      1.408      0.159      -0.011
0.068
================================================================================
=========
```

Differences and Findings Interestingly, this is the only regression output that shows a positive relationship between the number of spells used and winning. For higher level players, more flexibility is offered as a result.

Summary: Limitations to findings: This data was pulled strictly from 4 days of Clash Royale play in April 2023. Due to the limited data available, this was the best data that I could find in order to answer my question. The game has evolved since then and the game creators have changed attributes of certain cards, making certain cards worse and certain cards better by default. As a result, this makes finding the best cards an ever-evolving process.

In addition, this data was only compiled from matches involving Clash Royale's triple draft mode. The findings here abstract away the synergy that players can build between certain cards by allowing for the choices of cards to be as independent from one another as possible. Thus, while certain spells may contribute less to winning in draft, these cards may still be good choices in other facets of the game.

Future Improvements Due to memory / space constraints of my computer, I was unable to analyze more data. Clash Royale is very popular, as evident by the number of matches played in this 4 day span cracking 300,000. This is only data involving one mode of the game as well, which excludes the millions of other matches played in other modes of the game throughout this span.

In the future, I would like to extend this process out to include more recent data. In addition, I believe more features could've been controlled for when undergoing my logistic regression, such as the number of matches played from each player, as well as the attributes of the spells being analyzed. These variables are also important to determining who wins a Clash Royale match. However, that would require some scraping and API calls, neither of which I had the ability to do this time around.

mainmerge

May 9, 2024

0.1 Creating the Data: Econ 143 Final Project Sp24

- **includes sources for where the data came from, in addition to interpretations of all columns

[ ]:
```
import pandas as pd
import numpy as np
```

[ ]:
```
column_names = ["datetime", "gamemode", "p1tag", "p1trophies", "p1crowns",␣
↪"p1card1", "p1card2", "p1card3", "p1card4", "p1card5", "p1card6", "p1card7",␣
↪"p1card8", "p2tag", "p2trophies", "p2crowns", "p2card1", "p2card2",␣
↪"p2card3", "p2card4", "p2card5", "p2card6", "p2card7", "p2card8"]
```

Data Sources: * df_1, df_2, df_3, df_4: https://www.kaggle.com/datasets/s1m0n38/clash-royale-games * df_demog: https://www.kaggle.com/datasets/lucianomartins/data-game-clashroyale

[ ]:
```
# ALL Clash Royale Games played in 4/14, 4/15, 4/16, 4/17 of 2023

df_1 = pd.read_csv("20230414.csv", names = column_names)
df_2 = pd.read_csv("20230415.csv", names = column_names)
df_3 = pd.read_csv("20230416.csv", names = column_names)
df_4 = pd.read_csv("20230417.csv", names = column_names)
```

[ ]:
```
df = pd.concat([df_1, df_2, df_3, df_4])  # Combining all the data into 1␣
↪dataframe
df

'''Interpretation of Columns
* datetime: time the game ended
* gamemode: index corresponding to a different gamemode. Certain numbers␣
↪correspond to draft, which is the mode I wanted to analyze.

Player-specific variables: (p1 and p2)
* trophies: The number of trophies the player had before playing the match.
* crowns: The number of crowns a player won.
* p1card 1-8: The 8 unique cards that a player picked as their deck.

Explanations for the indexes can be found on https://www.kaggle.com/datasets/
↪s1m0n38/clash-royale-games.
'''
```

```
[ ]:                  datetime  gamemode      p1tag  p1trophies  p1crowns  \
0       20230414T000000.000Z  72000006  L8YYVY9UY        7500         0
1       20230414T000000.000Z  72000333   82UCRRL9          30         1
2       20230414T000000.000Z  72000333  98890C8GQ           0         0
3       20230414T000000.000Z  72000333  992JPJCU9           0         0
4       20230414T000000.000Z  72000333  PRGPPY2QV           0         0
…                          …         …          …         …        …
786379  20230417T235959.000Z  72000333  8Y9RCRJPQ           0         0
786380  20230417T235959.000Z  72000333   GGYYLP2L           0         0
786381  20230417T235959.000Z  72000333    GRQCJ99          30         2
786382  20230417T235959.000Z  72000333   PVPJPQQG           0         0
786383  20230417T235959.000Z  72000333   PVRCYL2P           0         0

         p1card1   p1card2   p1card3   p1card4   p1card5  …  p2trophies  \
0       26000015  26000023  26000027  26000085  27000012  …        7500
1       26000007  26000016  26000034  26000044  26000080  …           0
2       26000007  26000012  26000021  26000053  27000003  …          30
3       26000020  26000021  26000054  26000072  26000087  …          30
4       26000011  26000015  26000026  26000047  26000062  …          30
…              …         …         …         …         …  …           …
786379  26000028  26000045  26000053  26000077  27000010  …          30
786380  26000001  26000010  26000013  26000028  26000030  …          30
786381  26000005  26000012  26000025  26000032  26000053  …           0
786382  26000007  26000015  26000025  26000027  26000049  …          30
786383  26000001  26000013  26000032  26000072  27000005  …          30

        p2crowns   p2card1   p2card2   p2card3   p2card4   p2card5   p2card6  \
0              3  26000004  26000027  26000042  26000046  26000051  26000083
1              0  26000006  26000021  26000025  26000074  27000012  28000000
2              3  26000002  26000006  26000028  26000031  26000040  26000069
3              1  26000007  26000011  26000019  26000025  26000050  26000053
4              1  26000030  26000044  26000051  26000074  26000085  28000014
…              …         …         …         …         …         …         …
786379         1  26000011  26000018  26000041  26000056  26000057  26000062
786380         1  26000005  26000009  26000016  26000043  26000083  28000007
786381         0  26000016  26000018  26000024  26000043  26000051  28000000
786382         1  26000018  26000020  26000024  26000045  26000072  27000012
786383         3  26000007  26000016  26000021  26000042  26000044  26000051

         p2card7   p2card8
0       28000000  28000001
1       28000011  28000012
2       27000004  28000009
3       26000069  28000003
4       28000016  28000018
…              …         …
786379  26000072  28000008
```

```
786380  28000008  28000015
786381  28000005  28000009
786382  28000007  28000017
786383  28000001  28000003

[3168782 rows x 24 columns]
```

[ ]:
```
# draft_ids are the indexes corresponding to draft. I isolated the games that␣
↪were draft games.

draft_ids = [72000042, 72000333, 72000345, 72000346, 72000201, 72000003,␣
↪72000005, 72000013, 72000017, 72000022, 72000030, 72000040, 72000050,␣
↪72000051, 72000092, 72000106, 72000110, 72000111, 72000114, 72000115,␣
↪72000122, 72000135, 72000137, 72000143, 72000144, 72000153, 72000155,␣
↪72000160, 72000162, 72000167, 72000169, 72000175, 72000177, 72000182,␣
↪72000184, 72000188, 72000194, 72000348]
df_draft = df[df["gamemode"].isin(draft_ids)]
df_draft
```

[ ]:
```
                      datetime  gamemode      p1tag  p1trophies  p1crowns  \
1       20230414T000000.000Z  72000333   82UCRRL9          30         1
2       20230414T000000.000Z  72000333  98890C8GQ           0         0
3       20230414T000000.000Z  72000333  992JPJCU9           0         0
4       20230414T000000.000Z  72000333  PRGPPY2QV           0         0
8       20230414T000001.000Z  72000333  9GUQJ9LJQ           0         0
…                        …       …         …         …        …
786379  20230417T235959.000Z  72000333  8Y9RCRJPQ           0         0
786380  20230417T235959.000Z  72000333   GGYYLP2L           0         0
786381  20230417T235959.000Z  72000333    GRQCJ99          30         2
786382  20230417T235959.000Z  72000333   PVPJPQQG           0         0
786383  20230417T235959.000Z  72000333   PVRCYL2P           0         0

          p1card1   p1card2   p1card3   p1card4   p1card5  …  p2trophies  \
1        26000007  26000016  26000034  26000044  26000080  …           0
2        26000007  26000012  26000021  26000053  27000003  …          30
3        26000020  26000021  26000054  26000072  26000087  …          30
4        26000011  26000015  26000026  26000047  26000062  …          30
8        26000018  26000019  26000027  26000042  26000054  …          30
…             …       …       …       …        …  …         …
786379   26000028  26000045  26000053  26000077  27000010  …          30
786380   26000001  26000010  26000013  26000028  26000030  …          30
786381   26000005  26000012  26000025  26000032  26000053  …           0
786382   26000007  26000015  26000025  26000027  26000049  …          30
786383   26000001  26000013  26000032  26000072  27000005  …          30

         p2crowns   p2card1   p2card2   p2card3   p2card4   p2card5   p2card6  \
1               0  26000006  26000021  26000025  26000074  27000012  28000000
```

```
2              3  26000002  26000006  26000028  26000031  26000040  26000069
3              1  26000007  26000011  26000019  26000025  26000050  26000053
4              1  26000030  26000044  26000051  26000074  26000085  28000014
8              1  26000002  26000044  26000055  26000056  26000063  27000005
…          …      …         …         …         …         …         …
786379         1  26000011  26000018  26000041  26000056  26000057  26000062
786380         1  26000005  26000009  26000016  26000043  26000083  28000007
786381         0  26000016  26000018  26000024  26000043  26000051  28000000
786382         1  26000018  26000020  26000024  26000045  26000072  27000012
786383         3  26000007  26000016  26000021  26000042  26000044  26000051

         p2card7   p2card8
1       28000011  28000012
2       27000004  28000009
3       26000069  28000003
4       28000016  28000018
8       28000009  28000018
…          …         …
786379  26000072  28000008
786380  28000008  28000015
786381  28000005  28000009
786382  28000007  28000017
786383  28000001  28000003

[1785081 rows x 24 columns]
```

[ ]:

```python
# Altering the data so that these two columns display whether player1 won (1)␣
↪and whether player2 won (1).

df_draft["p1result"] = np.where(df_draft['p1crowns'] > df_draft['p2crowns'], 1,␣
↪0)
df_draft["p2result"] = np.where(df_draft['p2crowns'] > df_draft['p1crowns'], 1,␣
↪0)
```

```
/var/folders/k5/frt5sy_s1ng6fxz32_4sqkc00000gn/T/ipykernel_42343/1648582993.py:1
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_draft["p1result"] = np.where(df_draft['p1crowns'] > df_draft['p2crowns'],
1, 0)
/var/folders/k5/frt5sy_s1ng6fxz32_4sqkc00000gn/T/ipykernel_42343/1648582993.py:2
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_draft["p2result"] = np.where(df_draft['p2crowns'] > df_draft['p1crowns'],
1, 0)
```

[ ]:
```
df_draft
```

[ ]:
```
                     datetime  gamemode      p1tag  p1trophies  p1crowns  \
1       20230414T000000.000Z  72000333   82UCRRL9          30         1
2       20230414T000000.000Z  72000333  98890C8GQ           0         0
3       20230414T000000.000Z  72000333  992JPJCU9           0         0
4       20230414T000000.000Z  72000333  PRGPPY2QV           0         0
8       20230414T000001.000Z  72000333  9GUQJ9LJQ           0         0
…                        …       …         …         …       …
786379  20230417T235959.000Z  72000333  8Y9RCRJPQ           0         0
786380  20230417T235959.000Z  72000333   GGYYLP2L           0         0
786381  20230417T235959.000Z  72000333    GRQCJ99          30         2
786382  20230417T235959.000Z  72000333   PVPJPQQG           0         0
786383  20230417T235959.000Z  72000333   PVRCYL2P           0         0

         p1card1   p1card2   p1card3   p1card4   p1card5  …   p2card1  \
1       26000007  26000016  26000034  26000044  26000080  …  26000006
2       26000007  26000012  26000021  26000053  27000003  …  26000002
3       26000020  26000021  26000054  26000072  26000087  …  26000007
4       26000011  26000015  26000026  26000047  26000062  …  26000030
8       26000018  26000019  26000027  26000042  26000054  …  26000002
…            …       …       …       …       …  …       …
786379  26000028  26000045  26000053  26000077  27000010  …  26000011
786380  26000001  26000010  26000013  26000028  26000030  …  26000005
786381  26000005  26000012  26000025  26000032  26000053  …  26000016
786382  26000007  26000015  26000025  26000027  26000049  …  26000018
786383  26000001  26000013  26000032  26000072  27000005  …  26000007

         p2card2   p2card3   p2card4   p2card5   p2card6   p2card7   p2card8  \
1       26000021  26000025  26000074  27000012  28000000  28000011  28000012
2       26000006  26000028  26000031  26000040  26000069  27000004  28000009
3       26000011  26000019  26000025  26000050  26000053  26000069  28000003
4       26000044  26000051  26000074  26000085  28000014  28000016  28000018
8       26000044  26000055  26000056  26000063  27000005  28000009  28000018
…            …       …       …       …       …       …       …
786379  26000018  26000041  26000056  26000057  26000062  26000072  28000008
786380  26000009  26000016  26000043  26000083  28000007  28000008  28000015
786381  26000018  26000024  26000043  26000051  28000000  28000005  28000009
786382  26000020  26000024  26000045  26000072  27000012  28000007  28000017
786383  26000016  26000021  26000042  26000044  26000051  28000001  28000003

        p1result  p2result
```

```
1                 1           0
2                 0           1
3                 0           1
4                 0           1
8                 0           1
…             …         …
786379            0           1
786380            0           1
786381            1           0
786382            0           1
786383            0           1

[1785081 rows x 26 columns]
```

[ ]:
```
# Combining the information for both players, into 1 dataframe. Such that each␣
↪row of a dataframe represents 1 player and their results.

df_draft1 = df_draft[["datetime", "gamemode", "p1tag", "p1trophies",␣
↪"p1crowns", "p1card1", "p1card2", "p1card3", "p1card4", "p1card5",␣
↪"p1card6", "p1card7", "p1card8", "p1result"]]
df_draft2 = df_draft[["datetime", "gamemode", "p2tag", "p2trophies",␣
↪"p2crowns", "p2card1", "p2card2", "p2card3", "p2card4", "p2card5",␣
↪"p2card6", "p2card7", "p2card8", "p2result"]]

df_draft1.columns = ["datetime", "gamemode", "ptag", "ptrophies", "pcrowns",␣
↪"pcard1", "pcard2", "pcard3", "pcard4", "pcard5", "pcard6", "pcard7",␣
↪"pcard8", "presult"]
df_draft2.columns = ["datetime", "gamemode", "ptag", "ptrophies", "pcrowns",␣
↪"pcard1", "pcard2", "pcard3", "pcard4", "pcard5", "pcard6", "pcard7",␣
↪"pcard8", "presult"]
```

[ ]:
```
# Combining the dataframes
df_draft_revised = pd.concat([df_draft1, df_draft2])
```

[ ]:
```
df_draft_revised
```

[ ]:
```
                    datetime  gamemode       ptag  ptrophies  pcrowns  \
1        20230414T000000.000Z  72000333   82UCRRL9         30        1
2        20230414T000000.000Z  72000333  98890C8GQ          0        0
3        20230414T000000.000Z  72000333   992JPJCU9          0        0
4        20230414T000000.000Z  72000333  PRGPPY2QV          0        0
8        20230414T000001.000Z  72000333  9GUQJ9LJQ          0        0
…                        …         …          …         …        …
786379   20230417T235959.000Z  72000333  2C800VPCG         30        1
786380   20230417T235959.000Z  72000333   8Y92R880         30        1
786381   20230417T235959.000Z  72000333  GPPRCUY2J          0        0
786382   20230417T235959.000Z  72000333  2U928UCYL         30        1
```

```
786383  20230417T235959.000Z  72000333  P820QVLRQ        30        3

            pcard1    pcard2    pcard3    pcard4    pcard5    pcard6    pcard7  \
1         26000007  26000016  26000034  26000044  26000080  27000010  28000001
2         26000007  26000012  26000021  26000053  27000003  28000005  28000008
3         26000020  26000021  26000054  26000072  26000087  27000000  28000009
4         26000011  26000015  26000026  26000047  26000062  26000077  27000001
8         26000018  26000019  26000027  26000042  26000054  26000059  27000000
…              …         …         …         …         …         …         …
786379    26000011  26000018  26000041  26000056  26000057  26000062  26000072
786380    26000005  26000009  26000016  26000043  26000083  28000007  28000008
786381    26000016  26000018  26000024  26000043  26000051  28000000  28000005
786382    26000018  26000020  26000024  26000045  26000072  27000012  28000007
786383    26000007  26000016  26000021  26000042  26000044  26000051  28000001

            pcard8  presult
1         28000009        1
2         28000014        0
3         28000015        0
4         28000011        0
8         28000011        0
…              …       …
786379    28000008        1
786380    28000015        1
786381    28000009        0
786382    28000017        1
786383    28000003        1

[3570162 rows x 14 columns]
```

[ ]:

```
# Creating boolean variables, whether certain spells are part of a player's
↪deck.

df_draft_revised['fireball_bool'] = df_draft_revised[["pcard1", "pcard2",
↪"pcard3", "pcard4", "pcard5", "pcard6", "pcard7", "pcard8"]].eq(28000000).
↪any(axis=1).astype(int)
df_draft_revised['arrows_bool'] = df_draft_revised[["pcard1", "pcard2",
↪"pcard3", "pcard4", "pcard5", "pcard6", "pcard7", "pcard8"]].eq(28000001).
↪any(axis=1).astype(int)
df_draft_revised['rage_bool'] = df_draft_revised[["pcard1", "pcard2", "pcard3",
↪"pcard4", "pcard5", "pcard6", "pcard7", "pcard8"]].eq(28000002).any(axis=1).
↪astype(int)
df_draft_revised['rocket_bool'] = df_draft_revised[["pcard1", "pcard2",
↪"pcard3", "pcard4", "pcard5", "pcard6", "pcard7", "pcard8"]].eq(28000003).
↪any(axis=1).astype(int)
```

```python
df_draft_revised['freeze_bool'] = df_draft_revised[["pcard1", "pcard2", "pcard3", "pcard4", "pcard5", "pcard6", "pcard7", "pcard8"]].eq(28000005).any(axis=1).astype(int)
df_draft_revised['lightning_bool'] = df_draft_revised[["pcard1", "pcard2", "pcard3", "pcard4", "pcard5", "pcard6", "pcard7", "pcard8"]].eq(28000007).any(axis=1).astype(int)
df_draft_revised['zap_bool'] = df_draft_revised[["pcard1", "pcard2", "pcard3", "pcard4", "pcard5", "pcard6", "pcard7", "pcard8"]].eq(28000008).any(axis=1).astype(int)
df_draft_revised['poison_bool'] = df_draft_revised[["pcard1", "pcard2", "pcard3", "pcard4", "pcard5", "pcard6", "pcard7", "pcard8"]].eq(28000009).any(axis=1).astype(int)
df_draft_revised['the_log_bool'] = df_draft_revised[["pcard1", "pcard2", "pcard3", "pcard4", "pcard5", "pcard6", "pcard7", "pcard8"]].eq(28000011).any(axis=1).astype(int)
df_draft_revised['tornado_bool'] = df_draft_revised[["pcard1", "pcard2", "pcard3", "pcard4", "pcard5", "pcard6", "pcard7", "pcard8"]].eq(28000012).any(axis=1).astype(int)
df_draft_revised['earthquake_bool'] = df_draft_revised[["pcard1", "pcard2", "pcard3", "pcard4", "pcard5", "pcard6", "pcard7", "pcard8"]].eq(28000014).any(axis=1).astype(int)
df_draft_revised['barbarian_barrel_bool'] = df_draft_revised[["pcard1", "pcard2", "pcard3", "pcard4", "pcard5", "pcard6", "pcard7", "pcard8"]].eq(28000015).any(axis=1).astype(int)
df_draft_revised['snowball_bool'] = df_draft_revised[["pcard1", "pcard2", "pcard3", "pcard4", "pcard5", "pcard6", "pcard7", "pcard8"]].eq(28000017).any(axis=1).astype(int)
df_draft_revised['royal_delivery_bool'] = df_draft_revised[["pcard1", "pcard2", "pcard3", "pcard4", "pcard5", "pcard6", "pcard7", "pcard8"]].eq(28000018).any(axis=1).astype(int)
```

[ ]:
```python
df_draft_revised
```

[ ]:
```
                    datetime  gamemode       ptag  ptrophies  pcrowns  \
1       20230414T000000.000Z  72000333   82UCRRL9         30        1
2       20230414T000000.000Z  72000333  98890C8GQ          0        0
3       20230414T000000.000Z  72000333  992JPJCU9          0        0
4       20230414T000000.000Z  72000333  PRGPPY2QV          0        0
8       20230414T000001.000Z  72000333  9GUQJ9LJQ          0        0
…                          …         …          …          …        …
786379  20230417T235959.000Z  72000333  2C800VPCG         30        1
786380  20230417T235959.000Z  72000333   8Y92R880         30        1
786381  20230417T235959.000Z  72000333  GPPRCUY2J          0        0
786382  20230417T235959.000Z  72000333  2U928UCYL         30        1
786383  20230417T235959.000Z  72000333  P820QVLRQ         30        3
```

```
        pcard1    pcard2    pcard3    pcard4    pcard5  …  freeze_bool  \
1       26000007  26000016  26000034  26000044  26000080  …            0
2       26000007  26000012  26000021  26000053  27000003  …            1
3       26000020  26000021  26000054  26000072  26000087  …            0
4       26000011  26000015  26000026  26000047  26000062  …            0
8       26000018  26000019  26000027  26000042  26000054  …            0
…            …         …         …         …         …  …          …
786379  26000011  26000018  26000041  26000056  26000057  …            0
786380  26000005  26000009  26000016  26000043  26000083  …            0
786381  26000016  26000018  26000024  26000043  26000051  …            1
786382  26000018  26000020  26000024  26000045  26000072  …            0
786383  26000007  26000016  26000021  26000042  26000044  …            0

        lightning_bool  zap_bool  poison_bool  the_log_bool  tornado_bool  \
1                    0         0            1             0             0
2                    0         1            0             0             0
3                    0         0            1             0             0
4                    0         0            0             1             0
8                    0         0            0             1             0
…                  …         …            …             …             …
786379               0         1            0             0             0
786380               1         1            0             0             0
786381               0         0            1             0             0
786382               1         0            0             0             0
786383               0         0            0             0             0

        earthquake_bool  barbarian_barrel_bool  snowball_bool  \
1                     0                      0              0
2                     1                      0              0
3                     0                      1              0
4                     0                      0              0
8                     0                      0              0
…                   …                    …              …
786379                0                      0              0
786380                0                      1              0
786381                0                      0              0
786382                0                      0              1
786383                0                      0              0

        royal_delivery_bool
1                         0
2                         0
3                         0
4                         0
8                         0
…                       …
786379                    0
```

```
786380                    0
786381                    0
786382                    0
786383                    0

[3570162 rows x 28 columns]
```

[ ]:
```
# Taking the columns we care about
df_draft_rev_select = df_draft_revised[["presult", "ptag", "fireball_bool",␣
↪"arrows_bool", "rage_bool", "rocket_bool", "freeze_bool", "lightning_bool",␣
↪"zap_bool", "poison_bool", "the_log_bool", "tornado_bool",␣
↪"earthquake_bool", "barbarian_barrel_bool", "snowball_bool",␣
↪"royal_delivery_bool"]]
```

[ ]:
```
df_draft_rev_select
```

[ ]:
```
        presult       ptag  fireball_bool  arrows_bool  rage_bool  \
1             1   82UCRRL9              0            1          0
2             0  98890C8GQ              0            0          0
3             0  992JPJCU9              0            0          0
4             0  PRGPPY2QV              0            0          0
8             0  9GUQJ9LJQ              0            0          0
…           …         …             …           …         …
786379        1  2C800VPCG              0            0          0
786380        1   8Y92R880              0            0          0
786381        0  GPPRCUY2J              1            0          0
786382        1  2U928UCYL              0            0          0
786383        1  P820QVLRQ              0            1          0

        rocket_bool  freeze_bool  lightning_bool  zap_bool  poison_bool  \
1                 0            0               0         0            1
2                 0            1               0         1            0
3                 0            0               0         0            1
4                 0            0               0         0            0
8                 0            0               0         0            0
…               …            …              …         …           …
786379            0            0               0         1            0
786380            0            0               1         1            0
786381            0            1               0         0            1
786382            0            0               1         0            0
786383            1            0               0         0            0

        the_log_bool  tornado_bool  earthquake_bool  barbarian_barrel_bool  \
1                  0             0                0                      0
2                  0             0                1                      0
3                  0             0                0                      1
4                  1             0                0                      0
```

```
8                  1            0                0                      0
…               …            …               …                     …
786379             0            0                0                      0
786380             0            0                0                      0
786381             0            0                0                      1
786382             0            0                0                      0
786383             0            0                0                      0

        snowball_bool  royal_delivery_bool
1                   0                    0
2                   0                    0
3                   0                    0
4                   0                    0
8                   0                    0
…                 …                    …
786379              0                    0
786380              0                    0
786381              0                    0
786382              1                    0
786383              0                    0

[3570162 rows x 16 columns]
```

[ ]:

```python
# Demographic data

df_demog = pd.read_csv("players.csv")
```

[ ]:

```python
df_demog
```

[ ]:

```
          name_pais            name         tag  rank  expLevel  trophies  \
0       Afghanistan            f34r  #2UQ99VLU2     1        13      6148
1       Afghanistan      Ķiňğ Ãìmãĺ    #QJRU92L     2        13      6005
2       Afghanistan   L I Y A M  #800CLLPU8     3         13       5988
3       Afghanistan  FRAIDOON ISHAQ    #9LLLJP8C     4        13      5920
4       Afghanistan    king behzad   #P9VLCU8Y9     5        13      5912
…             …             …          …    …       …        …
167228     Zimbabwe      SethyPlayz  #QQ2U8JCP8   397         2        30
167229     Zimbabwe        kingkong  #R0G2RYCJ2   398         2        30
167230     Zimbabwe   Lord Gargamel  #Y8Y2Y0PV0   399         2        29
167231     Zimbabwe          mattm8  #J089J0YCG   400         2        29
167232     Zimbabwe          sipoko  #R2U8RG0R8   401         2        28

          name_clan   tag_clan  id_arena      name_arena
0          PhatKatz    #G9LGQR  54000015        Master I
1        God of War   #V2RUJ88  54000015        Master I
2                #L2CC2L9C  54000015        Master I
3       Afghanistan  #PGV80J0P  54000014  Challenger III
```

```
4       Delta Force  #LR8P9L8C  54000014  Challenger III
…                …         …         …               …
167228           NaN         NaN  54000001         Arena 1
167229           NaN         NaN  54000001         Arena 1
167230           NaN         NaN  54000001         Arena 1
167231          join  #YRUUYOCO  54000001         Arena 1
167232           NaN         NaN  54000001         Arena 1

[167233 rows x 10 columns]
```

[ ]:
```
# Removing the # in the 'tag' column, in order to merge dataframes by player id
df_demog['ptag'] = df_demog['tag'].str.replace('#', '')
```

[ ]:
```
df_demog
```

[ ]:
```
          name_pais            name        tag  rank  expLevel  trophies  \
0       Afghanistan            f34r  #2UQ99VLU2     1        13      6148
1       Afghanistan      Ķiňğ Ãìmãĺ    #QJRU92L     2        13      6005
2       Afghanistan    L I Y A M  #800CLLPU8      3        13      5988
3       Afghanistan  FRAIDOON ISHAQ   #9LLLJP8C     4        13      5920
4       Afghanistan    king behzad  #P9VLCU8Y9     5        13      5912
…                …              …          …   …        …         …
167228     Zimbabwe      SethyPlayz  #QQ2U8JCP8   397         2        30
167229     Zimbabwe        kingkong  #R0G2RYCJ2   398         2        30
167230     Zimbabwe  Lord Gargamel  #Y8Y2Y0PV0   399         2        29
167231     Zimbabwe          mattm8  #J089J0YCG   400         2        29
167232     Zimbabwe          sipoko  #R2U8RG0R8   401         2        28

          name_clan   tag_clan  id_arena      name_arena       ptag
0          PhatKatz   #G9LGQR  54000015        Master I  2UQ99VLU2
1        God of War   #V2RUJ88  54000015        Master I    QJRU92L
2               #L2CC2L9C  54000015        Master I  800CLLPU8
3       Afghanistan  #PGV80J0P  54000014  Challenger III   9LLLJP8C
4       Delta Force  #LR8P9L8C  54000014  Challenger III  P9VLCU8Y9
…                …         …         …               …          …
167228           NaN         NaN  54000001         Arena 1  QQ2U8JCP8
167229           NaN         NaN  54000001         Arena 1  R0G2RYCJ2
167230           NaN         NaN  54000001         Arena 1  Y8Y2Y0PV0
167231          join  #YRUUYOCO  54000001         Arena 1  J089J0YCG
167232           NaN         NaN  54000001         Arena 1  R2U8RG0R8

[167233 rows x 11 columns]
```

[ ]:
```
# Merging our previous dataframe with demographic data

df_merged = pd.merge(df_draft_rev_select, df_demog[["ptag", "expLevel",␣
 ↪"trophies"]], on = "ptag", how = "inner")
```

[ ]:
```
df_merged
```

[ ]:
```
        presult       ptag  fireball_bool  arrows_bool  rage_bool  \
0             0   PRQJRGJ8              0            0          0
1             1  P2C2CCQ2V              0            0          0
2             1  9JVCJ92V9              1            0          0
3             1  Y82RVYPC8              0            1          1
4             1   Y89UL2JL              0            0          0
…           …        …            …          …        …
306241        0  22GL0VJQJ              0            1          0
306242        1  2LJJQYJ8J              0            0          0
306243        1  2RRV98JU8              0            0          0
306244        0  2PQ8VGCUL              1            0          0
306245        1  2899RCU28              0            0          0

        rocket_bool  freeze_bool  lightning_bool  zap_bool  poison_bool  \
0                 0            0               0         0            0
1                 0            0               0         0            0
2                 0            0               0         0            0
3                 0            0               0         0            1
4                 0            0               0         0            1
…             …          …             …       …          …
306241            0            0               0         0            0
306242            0            0               0         1            0
306243            0            0               1         1            0
306244            0            0               0         0            0
306245            0            0               0         0            1

        the_log_bool  tornado_bool  earthquake_bool  barbarian_barrel_bool  \
0                  0             0                0                      0
1                  0             0                0                      1
2                  1             0                0                      0
3                  0             0                0                      0
4                  0             0                0                      0
…              …           …              …                    …
306241             1             0                1                      0
306242             0             0                0                      0
306243             0             0                0                      1
306244             0             0                0                      1
306245             0             0                1                      1

        snowball_bool  royal_delivery_bool  expLevel  trophies
0                   0                    0        13      5518
1                   0                    0        13      6487
2                   0                    0        13      6600
3                   0                    0        13      5695
4                   0                    0        13      5885
```

```
…                …                   …       …       …
306241              0                     0       13       5987
306242              0                     0       13       6005
306243              1                     0       13       5793
306244              0                     0       13       5924
306245              0                     0       13       6082

[306246 rows x 18 columns]
```

[ ]:

```
# Exporting the data, which is used in Econ143.ipynb. for analysis.

df_merged.to_csv("triple_draft_data.csv")
```