Documentation

Index:

Data Recollection

The data recollection started from the website https://aoe-elo.com/, the website's specific focus is to have a collection of all ao2 tournament game results, this means they have data from the first age of empires 2 tournaments until the tournament in July 2023. The data is structured in 14 tables, we could go into further detail about the contents of each table, but right now as a main focus we will talk about 5 tables which are the following:

- Tournament
    - We use tournament to keep track of the tournament name, level of the tournament (tier), amount of prize money related to each tournament
- Match_1v1
    - This table provides us directly the information on each game related to a specific tournament, this includes score between the two players and to which tournament this score corresponds to as well as identifying the stage_id which is just an indicator of the knockout stage (1 being a group stage and this number increases to show which part of the tournament we are in).
- Player, player_cache

    - Player and player_cache gives us information on each specific player in each match, this means it gives us the elo of each player, the rank of each player (rank can be calculated by the elo), the name, and id of each player.
- Stage
    - Stage indicates in which specific part of the tournament we are in, this could be group stage all the way up until the finale.

It is important to note that the data provided in the website contains all tournaments the beginning of age of empires 2, but not all the matches from each specific tournament since they specifically mention the following:

- We don't put admin wins or other canceled games in the database, only really played matches.
- Not all stages of all tournaments are important. If the early stages of a tournament only consist of mainly low-tier matches, we don't put them in the database. This way, the amount of low-level players with only a few registered matches is lower, and they do not clutter the statistics and tables too much.

As cited in the website Tournament elo (2023). So it is important to note even though the data within the website provides full detail and understanding of each tournament with its matches. The tournaments provided in the data do not have all the matches within the specific tournament within that database.

I obtained the data from the database through an sql file, the table I queried has the data from each specific match the columns were the following:

- Tournament name
- Tournament_id
- Prizemoney

- Knockout_stage
- Player_1_id
- Player_1_name
- Elo_player_1
- Score_1
- Player_2_id
- Player_2_name
- Elo_player_2
- score_2

I obtained the data using the mysql.connector library with the specific intent of just getting data on Hidden cup tournaments, the following query gives us the previous table format, using python to edit and clean the data to get the table in the proper format.

Query:

```
SELECT
    tournament.name,
    match_1v1.id,
    match_1v1.tournament_id,
    tournament.prizemoney,
    match_1v1.stage_id AS Knockout_stage,
    match_1v1.player_1_id,
    player1.name AS player_1_name,
    cache1.elo AS elo_player_1,
    match_1v1.score_1,
```

```
    match_1v1.player_2_id,
    player2.name AS player_2_name,
    cache2.elo AS elo_player_2,
    match_1v1.score_2
FROM
    tournament
INNER JOIN
    match_1v1 ON tournament.id = match_1v1.tournament_id
INNER JOIN
    player_cache AS cache1 ON match_1v1.player_1_id = cache1.player_id
INNER JOIN
    player AS player1 ON cache1.player_id = player1.id
INNER JOIN
    player_cache AS cache2 ON match_1v1.player_2_id = cache2.player_id
INNER JOIN
    player AS player2 ON cache2.player_id = player2.id
WHERE
    tournament.name LIKE '%Hidden Cup%';
```

The previous query gave us the data for only Hidden cup style type of tournaments, now we need a query that gives the same information on each match, but now for every other tournament.

Query:

```
SELECT
    tournament.name,
    match_1v1.id,
    match_1v1.tournament_id,
    tournament.prizemoney,
    match_1v1.stage_id AS Knockout_stage,
    match_1v1.player_1_id,
    player1.name AS player_1_name,
    cache1.elo AS elo_player_1,
    match_1v1.score_1,
```

```
    match_1v1.player_2_id,
    player2.name AS player_2_name,
    cache2.elo AS elo_player_2,
    match_1v1.score_2
FROM
    tournament
INNER JOIN
    match_1v1 ON tournament.id = match_1v1.tournament_id
INNER JOIN
    player_cache AS cache1 ON match_1v1.player_1_id =
cache1.player_id
INNER JOIN
    player AS player1 ON cache1.player_id = player1.id
INNER JOIN
    player_cache AS cache2 ON match_1v1.player_2_id =
cache2.player_id
INNER JOIN
    player AS player2 ON cache2.player_id = player2.id
WHERE
    tournament.name NOT LIKE '%Hidden Cup%';
```

Now, we can see this tables are disjoint:

A = Hidden cup matches info

B = Rest of the tournaments matches info

So $A \cap B = \phi$.

Given these tables we can begin the data cleaning, so first all the data I obtained does not contain null values since the queries are filtered based on inner join this means there will be no null values to all parameters of interest that cause a difference in the outcome of a match. Now we will add 2 extra columns which indicate the tier level of each tournament associated with each match and the underdogs.

Data Cleaning

We generate the tables with the following code:

```
#data on the database connection
cnx = mysql.connector.connect(host='localhost', user='root'
, password='mauricio02', database='age')

query = '''
SELECT
    tournament.name,
    match_1v1.id,
    match_1v1.tournament_id,
    tournament.prizemoney,
    match_1v1.stage_id AS Knockout_stage,
    match_1v1.player_1_id,
    player1.name AS player_1_name,
    cache1.elo AS elo_player_1,
    match_1v1.score_1,
    match_1v1.player_2_id,
    player2.name AS player_2_name,
    cache2.elo AS elo_player_2,
    match_1v1.score_2
FROM
    tournament
INNER JOIN
    match_1v1 ON tournament.id = match_1v1.tournament_id
INNER JOIN
    player_cache AS cache1 ON match_1v1.player_1_id = cache1.player_id
INNER JOIN
    player AS player1 ON cache1.player_id = player1.id
INNER JOIN
    player_cache AS cache2 ON match_1v1.player_2_id = cache2.player_id
INNER JOIN
    player AS player2 ON cache2.player_id = player2.id
WHERE
    tournament.name LIKE '%Hidden Cup%';
'''

query1 = '''
```

```
SELECT
    tournament.name,
    match_1v1.id,
    match_1v1.tournament_id,
    tournament.prizemoney,
    match_1v1.stage_id AS Knockout_stage,
    match_1v1.player_1_id,
    player1.name AS player_1_name,
    cache1.elo AS elo_player_1,
    match_1v1.score_1,
    match_1v1.player_2_id,
    player2.name AS player_2_name,
    cache2.elo AS elo_player_2,
    match_1v1.score_2
FROM
    tournament
INNER JOIN
    match_1v1 ON tournament.id = match_1v1.tournament_id
INNER JOIN
    player_cache AS cache1 ON match_1v1.player_1_id = cache1.player_id
INNER JOIN
    player AS player1 ON cache1.player_id = player1.id
INNER JOIN
    player_cache AS cache2 ON match_1v1.player_2_id = cache2.player_id
INNER JOIN
    player AS player2 ON cache2.player_id = player2.id
WHERE
    tournament.name NOT LIKE '%Hidden Cup%';
'''

dfh = pd.read_sql(query, cnx)
df = pd.read_sql(query1, cnx)

cnx.close()
```

First we will remove the qualifier tournaments in the hidden cup because they do not take into consideration the hidden aspect, with the following code.

Note: dfh indicates the dataframe with only hidden tournaments

Code:

```
filt = (dfh['name'].str.contains('Qualifier'))
dfh = dfh[~filt]
```

Given that we now add 2 columns for a Hidden aspect and to indicate whether the tournament is s-tier or not (all Hidden cup tournaments are s-tier).

Code:

```
dfh['s-tier'] = True
dfh['hidden'] = dfh['name'].str.contains('Hidden Cup')
```

Sample data frame:

	name	id	tournament_id	prizemoney	Knockout_stage	player_1_id	player_1_name	elo_player_1	score_1	player_2_id	player_2_name	elo_player_2	score_2	s-tier	hidden
0	Hidden Cup	222	7	1440.0	2	1	DauT	2286	1	2	miguel	2169	3	True	True
1	Hidden Cup	223	7	1440.0	2	27	Nicov	2288	3	31	TaToH	2369	1	True	True
2	Hidden Cup	224	7	1440.0	2	19	Yo	2354	2	3	RiuT	2155	3	True	True
3	Hidden Cup	225	7	1440.0	2	5	F1Re	2146	3	35	vivi	2234	1	True	True
4	Hidden Cup	226	7	1440.0	4	25	Liereyy	2398	2	2	miguel	2169	3	True	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
153	Hidden Cup Italia	5236	284	947.0	4	619	Rise	2006	3	989	Zuccolo	1958	0	True	True
154	Hidden Cup Italia	5237	284	947.0	5	129	Vodka_L	2113	3	503	pete_martell	1989	1	True	True
155	Hidden Cup Italia	5238	284	947.0	5	446	Kamigawa	1983	2	619	Rise	2006	3	True	True
156	Hidden Cup Italia	5239	284	947.0	12	503	pete_martell	1989	2	446	Kamigawa	1983	3	True	True
157	Hidden Cup Italia	5240	284	947.0	6	129	Vodka_L	2113	4	619	Rise	2006	0	True	True

Data Analysis:

In the following section we obtain summary statistics for tournaments, players, and underdogs.

We first have to define what an underdog is. Elo is a method to measure player strength. So by seeing a difference in elo we can also see a difference in player level and skill, so by taking into account the elo we can notice who is better and who isn't.

An "underdog win" occurs in a match when a player with a lower Elo rating defeats their opponent with a higher Elo rating. This situation often represents a surprising or unexpected outcome, where the perceived weaker player triumphs over the stronger one based on their respective Elo ratings.

So there can be different types of underdogs, we can categorize an underdog when the difference in elo is 10,20,50,100.

The following function checks if the underdog wins a certain match, the threshold parameter takes into account the difference elo there is for there to be an underdog.

Code:

```
def checkifUnderdogwins(row, threshold):
   if abs(row['elo_player_1']-row['elo_player_2']) > threshold:
       if min(row['elo_player_1'],row['elo_player_2']) == row['elo_player_1']:
           return row['score_1'] > row['score_2']
       else:
           return row['score_2'] > row['score_1']
   else:
       return False
```

Once taken into account the threshold we calculate each proportion of winrate of the underdogs in each specific threshold through the following function:

```
thresholds = [10,20,50,100]
def createplot(df,thresholds):
   data = []
   for threshold in thresholds:
       df['Underdog'] = df.apply(checkifUnderdogwins, axis=1, args=(threshold,))
       val = df['Underdog'].value_counts()
       false = val[False]
       true = val[True]
       arr = [true, false]
       data.append(arr)

   return data
```

This means after applying the function we have a specific percentage of win rate of underdogs given N matches.

Now that we have a function that determines our winrate, we split our data.

Through regex we can separate the following data into 3 separate data frames, we can use hidden, s-tier and normal tournament category. Now that we have data into 3 separate categories of tournaments, we can extract data and organize into 3 main branches:

- Normal
- Hidden
- S-tier

Summarized Statistics

Now given our 3 main dataframes we can obtain data from each particular tournament category. We can first obtain some general statistics through some general data analysis exploration

Data-

Some basic statistics from each given tournament was the prize money, the elo and the number of matches per tournament. It is important to take into account that in the database not all matches are registered, so this is just a sample, but the size of the tournament does go into proportion to the amount of matches registered in the database, this means the bigger the tournament the more matches are registered in the database.

Tournament:

Tier	Attribute	mean	std	min	25%	50%	75%	max
Normal	Prize_Money	2307.29	6386.2	0.0	300.0	860.0	2065.0	100000.0
	Elo	2082.26596	147.927476	1858.0	1965.0	2040.0	2179.0	2446.0
	Match_num	15.471311	17.352109	1.0	4.0	8.0	18.0	96.0
s-tier	Prize_Money	26366.9	32932.23	500.0	2400.0	17300.0	35000.0	200000.0
	Elo	2123.141716	158.050747	1858.0	1994.0	2121.0	2264.0	2446.0
	Match_num	49.890625	32.921801	4.0	27.0	44.0	60.0	155.0
hidden	Prize_Money	30306.45	38391.43	947.0	1440.0	10200.0	87240.0	87240.0
	Elo	2258.085938	135.996875	1894.0	2169.0	2286.0	2369.0	2446.0
	Match_num	12.8	3.193744	8.0	11.0	15.0	15.0	15.0

Index:

- Tier: indicates type of tournament
- Attribute: the specific attribute we are gaining data from

UnderDog win rate given each tournament:

Given each type of tournament category we can calculate the winrate of underdogs for each specific threshold. The columns indicate the threshold level so we can identify an underdog when the difference in elo is greater than the threshold, and the underdog wins the match.

Example:

When the threshold is 10 the winrate of underdogs in hidden tournaments is of 29.69%

This means the difference in elo is greater than 10 and underdogs win 29 percent of the time in hidden tournaments.

		10	20	50	100
Tier	Attribute				
Normal	Winrate	20.53%	18.37%	13.19%	7.44%
s-tier	Winrate	16.07%	13.53%	8.83%	5.01%
hidden	Winrate	29.69%	26.56%	20.31%	12.5%

Players:

From this small table we can get a small summary of player statistics in general, were we can show the number of wins per player and the general elo rating.

att	mean	std	min	25%	50%	75%	max
elo	1929.769341	70.857977	1858.0	1894.0	1898.0	1951.0	2446.0
wins	7.542264	28.761515	0.0	0.0	0.0	3.0	385.0