

PrettyFormatted_LOL_DATA

January 16, 2025

1 Let's see the data that we collected

First we will need to install pymongo

```
[1]: !pip install pymongo
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pymongo in
c:\users\gctan\appdata\roaming\python\python312\site-packages (4.10.1)
Requirement already satisfied: dnspython<3.0.0,>=1.16.0 in
c:\users\gctan\appdata\roaming\python\python312\site-packages (from pymongo)
(2.7.0)
```

1.1 Other imports:

- in order to use the mongo DB where we have our data stored, we will need **pymongo**
- **pandas** will allow us to manipulate the data and organize it for visualizations
- **matplotlib** and **seaborn** to display heatmap

```
[20]: import pymongo
from pymongo import MongoClient
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
import numpy as np
```

1.2 Connecting to the MongoDB DataSource:

```
[3]: client = MongoClient("mongodb://root:example@localhost:27017/")
try:
    # List all databases to check the connection
    print(client.list_database_names())
except Exception as e:
    print(f"Error: {e}")
```

```
['admin', 'config', 'local', 'lol_data']
```

The database that is of interest to us is `lol_data`. Inside it we will have the `match_info` collection that we will be playing with.

```
[4]: db = client["lol_data"]
match_info = db["match_info"]
result = match_info.count_documents({})
print(f"Number of documents in match_info collection: {result}")
```

Number of documents in match_info collection: 5345

The jsons saved, contain a lot of data, and the structure of the data is something like this:

- id
- metadata
 - some properties
 - participants - array of strings
- info
 - some properties
 - participants - array of objects
 - * some properties
 - * challenges - object
 - contains some properties
 - 1 array of ints
 - perks - object that contains arrays of objects
 - teams - an array with objects

I took some of the elements from the Collections that I thought would influence the win rate...

```
[5]: collection = db["match_info"]

data = {
    "gameId": [],
    "champLevel": [],
    "champExperience": [],
    "championId": [],
    "championName": [],
    "summonerLevel": [],
    "longestTimeSpentLiving": [],
    "totalTimeCCDealt": [],
    "totalTimeSpentDead": [],
    "skillshotsDodged": [],

    "damagePerMinute": [],
    "damageDealtToObjectives": [],
    "damageSelfMitigated": [],
    "trueDamageDealt": [],
    "trueDamageTaken": [],
    "largestCriticalStrike": [],
    "totalUnitsHealed": [],
    "trueDamageDealtToChampions": [],
    "magicDamageDealt": [],
    "magicDamageDealtToChampions": []
}
```

```

"physicalDamageDealt": [],
"physicalDamageDealtToChampions": [],
"physicalDamageTaken": [],
"magicDamageTaken": [],
"totalDamageDealt": [],
"totalDamageDealtToChampions": [],
"totalDamageTaken": [],
"totalHeal": [],

"kda": [],
"kills": [],
    "deaths": [],
"killingSprees": [],
"largestKillingSpree": [],
"largestMultiKill": [],
"multikills": [],
"doubleKills": [],
"tripleKills": [],
    "quadraKills": [],
"pentaKills": [],
"killsNearEnemyTurret": [],
"maxKillDeficit": [],

"timePlayed": [],
    "gameDuration": [],

"totalMinionsKilled": [],
"goldEarned": [],
"goldSpent": [],
"itemsPurchased": [],
"item0": [],
"item1": [],
"item2": [],
"item3": [],
"item4": [],
"item5": [],
"item6": [],
"spell1Casts": [],
"spell2Casts": [],
"spell3Casts": [],
"spell4Casts": [],

    "turretKills": [],
"turretTakedowns": [],
"turretsLost": [],
"inhibitorsLost": [],

```

```

    "teamId": [],
    "gameEndedInEarlySurrender": [],
    "gameEndedInSurrender": [],
    "win": [],

    "puuid": [],
    "riotIdGameName": [],
    "riotIdTagline": []
}

```

```

[6]: # Query all documents in the collection
for document in collection.find():
    # Extract participants from each document
    participants = document.get("info", {}).get("participants", [])
    for participant in participants:
        # General Game data:
        data["gameId"].append(document.get("info", {}).get("gameId"))
        data["gameDuration"].append(document.get("info", {}).
↳get("gameDuration"))
        # Participant - Challenges Area
        data["damagePerMinute"].append(participant.get("challenges").
↳get('damagePerMinute'))
        data["kda"].append(participant.get("challenges").get('kda'))
        data["killsNearEnemyTurret"].append(participant.get("challenges").
↳get('killsNearEnemyTurret'))
        data["maxKillDeficit"].append(participant.get("challenges").
↳get('maxKillDeficit'))
        data["multikills"].append(participant.get("challenges").
↳get('multikills'))
        data["skillshotsDodged"].append(participant.get("challenges").
↳get('skillshotsDodged'))
        # Participant
        data["champExperience"].append(participant.get('champExperience'))
        data["champLevel"].append(participant.get('champLevel'))
        data["championId"].append(participant.get('championId'))
        data["championName"].append(participant.get('championName'))
        data["damageDealtToObjectives"].append(participant.
↳get('damageDealtToObjectives'))
        data["damageSelfMitigated"].append(participant.
↳get('damageSelfMitigated'))
        data["deaths"].append(participant.get('deaths'))
        data["doubleKills"].append(participant.get('doubleKills'))
        data["gameEndedInEarlySurrender"].append(participant.
↳get('gameEndedInEarlySurrender'))

```

```

        data["gameEndedInSurrender"].append(participant.
↪get('gameEndedInSurrender'))
        data["goldEarned"].append(participant.get('goldEarned'))
        data["goldSpent"].append(participant.get('goldSpent'))
        data["inhibitorsLost"].append(participant.get('inhibitorsLost'))
        data["item0"].append(participant.get('item0'))
        data["item1"].append(participant.get('item1'))
        data["item2"].append(participant.get('item2'))
        data["item3"].append(participant.get('item3'))
        data["item4"].append(participant.get('item4'))
        data["item5"].append(participant.get('item5'))
        data["item6"].append(participant.get('item6'))
        data["itemsPurchased"].append(participant.get('itemsPurchased'))
        data["killingSprees"].append(participant.get('killingSprees'))
        data["kills"].append(participant.get('kills'))
        data["largestCriticalStrike"].append(participant.
↪get('largestCriticalStrike'))
        data["largestKillingSpree"].append(participant.
↪get('largestKillingSpree'))
        data["largestMultiKill"].append(participant.get('largestMultiKill'))
        data["longestTimeSpentLiving"].append(participant.
↪get('longestTimeSpentLiving'))
        data["magicDamageDealt"].append(participant.get('magicDamageDealt'))
        data["magicDamageDealtToChampions"].append(participant.
↪get('magicDamageDealtToChampions'))
        data["magicDamageTaken"].append(participant.get('magicDamageTaken'))
        data["quadraKills"].append(participant.get('quadraKills'))
        data["pentaKills"].append(participant.get('pentaKills'))
        data["physicalDamageDealt"].append(participant.
↪get('physicalDamageDealt'))
        data["physicalDamageDealtToChampions"].append(participant.
↪get('physicalDamageDealtToChampions'))
        data["physicalDamageTaken"].append(participant.
↪get('physicalDamageTaken'))
        data["riotIdGameName"].append(participant.get('riotIdGameName'))
        data["riotIdTagline"].append(participant.get('riotIdTagline'))
        data["puuid"].append(participant.get('puuid'))
        data["spell1Casts"].append(participant.get('spell1Casts'))
        data["spell2Casts"].append(participant.get('spell2Casts'))
        data["spell3Casts"].append(participant.get('spell3Casts'))
        data["spell4Casts"].append(participant.get('spell4Casts'))
        data["summonerLevel"].append(participant.get('summonerLevel'))
        data["teamId"].append(participant.get('teamId'))
        data["timePlayed"].append(participant.get('timePlayed'))
        data["totalDamageDealt"].append(participant.get('totalDamageDealt'))

```

```

        data["totalDamageDealtToChampions"].append(participant.
↪get('totalDamageDealtToChampions'))
        data["totalDamageTaken"].append(participant.get('totalDamageTaken'))
        data["totalHeal"].append(participant.get('totalHeal'))
        data["totalMinionsKilled"].append(participant.get('totalMinionsKilled'))
        data["totalTimeCCDealt"].append(participant.get('totalTimeCCDealt'))
        data["totalTimeSpentDead"].append(participant.get('totalTimeSpentDead'))
        data["totalUnitsHealed"].append(participant.get('totalUnitsHealed'))
        data["tripleKills"].append(participant.get('tripleKills'))
        data["trueDamageDealt"].append(participant.get('trueDamageDealt'))
        data["trueDamageDealtToChampions"].append(participant.
↪get('trueDamageDealtToChampions'))
        data["trueDamageTaken"].append(participant.get('trueDamageTaken'))
        data["turretKills"].append(participant.get('turretKills'))
        data["turretTakedowns"].append(participant.get('turretTakedowns'))
        data["turretsLost"].append(participant.get('turretsLost'))
        data["win"].append(participant.get('win'))

# DEBUG: see the amount of lines found for each:
# for x in data:
#     print(x, " ", len(data[x]))

```

```

[7]: onlyNumbers = pd.DataFrame.from_dict(data).drop(columns=['championName', '
↪gameId', 'riotIdGameName', 'riotIdTagline', 'puuid', 'teamId' ])

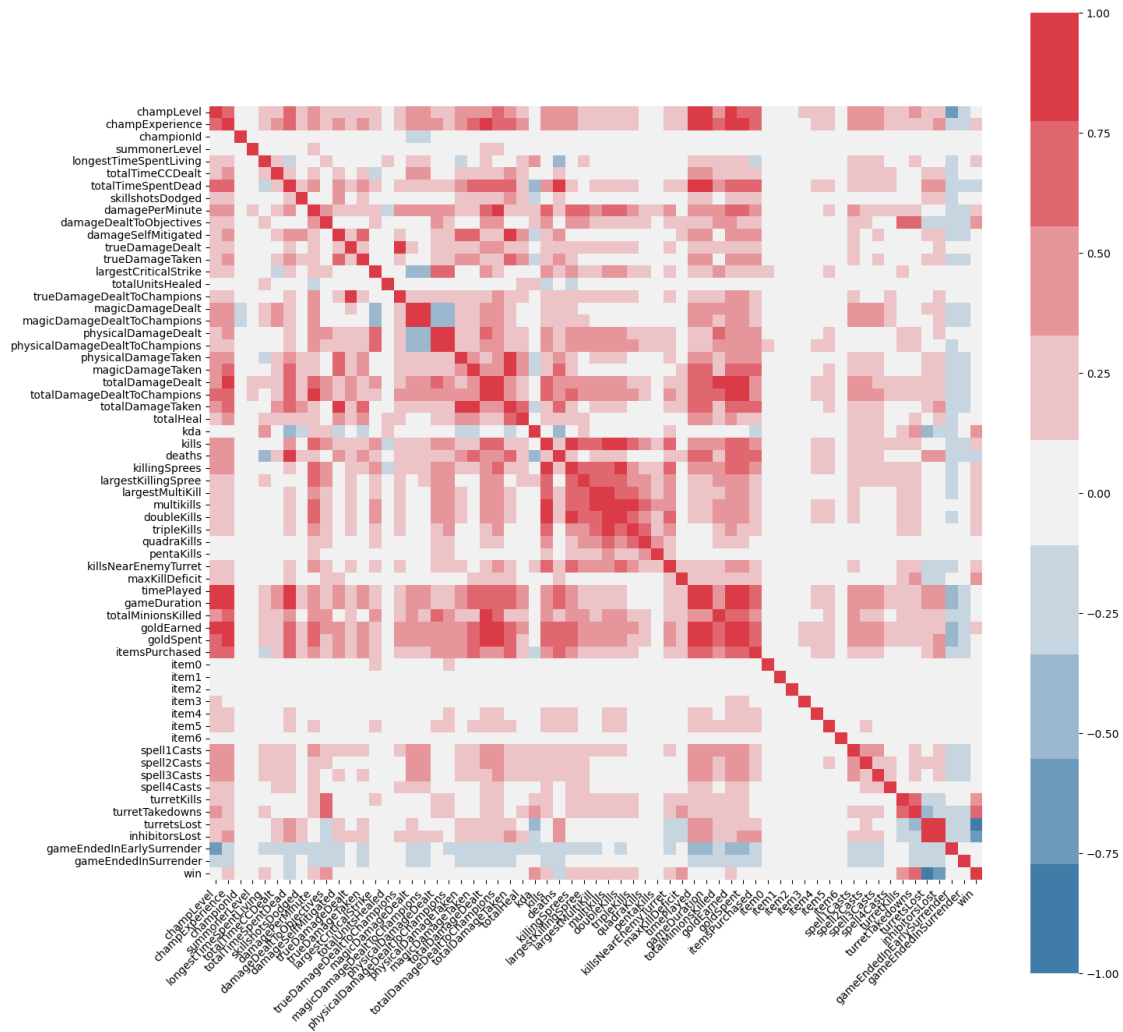
```

```

[8]: corr_data = onlyNumbers.corr()

plt.figure(figsize = (16,16))
ax_data = sns.heatmap(
    corr_data,
    vmin=-1,
    vmax=1,
    center=0,
    cmap=sns.diverging_palette(240, 10, n=9),
    square=True,
)
ax_data.set_xticklabels(
    ax_data.get_xticklabels(), rotation=45, horizontalalignment="right"
);

```



```
[9]: corr_data[(corr_data["win"] > 0.25) | (corr_data["win"] < -0.25)]["win"]
```

```
[9]: damageDealtToObjectives    0.463778
kda                            0.373734
killsNearEnemyTurret          0.307025
maxKillDeficit                0.484421
turretKills                   0.352146
turretTakedowns              0.617217
turretsLost                  -0.806603
inhibitorsLost               -0.641179
win                           1.000000
Name: win, dtype: float64
```

```
[10]: len(corr_data[(corr_data["timePlayed"] > 0.25) | (corr_data["timePlayed"] < -0.
↪25)]["timePlayed"])
```

```
[10]: 36
```

```
[11]: tags = [  
    "champLevel", "champExperience", "championId", "summonerLevel",  
    ↪ "longestTimeSpentLiving", "totalTimeCCDealt",  
    "totalTimeSpentDead", "skillshotsDodged", "damagePerMinute",  
    ↪ "damageDealtToObjectives", "damageSelfMitigated", "trueDamageDealt",  
    "trueDamageTaken", "largestCriticalStrike", "totalUnitsHealed",  
    ↪ "trueDamageDealtToChampions", "magicDamageDealt",  
    ↪ "magicDamageDealtToChampions",  
    "physicalDamageDealt", "physicalDamageDealtToChampions",  
    ↪ "physicalDamageTaken", "magicDamageTaken", "totalDamageDealt",  
    "totalDamageDealtToChampions", "totalDamageTaken", "totalHeal", "kda",  
    ↪ "kills", "deaths", "killingSpree", "largestKillingSpree",  
    "largestMultiKill", "multikills", "doubleKills", "tripleKills",  
    ↪ "quadraKills", "pentaKills", "killsNearEnemyTurret", "maxKillDeficit",  
    "timePlayed", "gameDuration", "totalMinionsKilled", "goldEarned",  
    ↪ "goldSpent", "itemsPurchased", "item0", "item1", "item2", "item3", "item4",  
    "item5", "item6", "spell1Casts", "spell2Casts", "spell3Casts",  
    ↪ "spell4Casts", "turretKills", "turretTakedowns", "turretsLost",  
    ↪ "inhibitorsLost",  
    "gameEndedInEarlySurrender", "gameEndedInSurrender", "win"]
```

```
[12]: for tag in tags:  
    print(tag, " => ", len(corr_data[(corr_data[tag] > 0.25) | (corr_data[tag]  
    ↪ < -0.25)][tag]))
```

```
champLevel => 36  
champExperience => 42  
championId => 1  
summonerLevel => 1  
longestTimeSpentLiving => 4  
totalTimeCCDealt => 15  
totalTimeSpentDead => 31  
skillshotsDodged => 10  
damagePerMinute => 32  
damageDealtToObjectives => 24  
damageSelfMitigated => 18  
trueDamageDealt => 11  
trueDamageTaken => 17  
largestCriticalStrike => 17  
totalUnitsHealed => 2  
trueDamageDealtToChampions => 16  
magicDamageDealt => 22  
magicDamageDealtToChampions => 20  
physicalDamageDealt => 28  
physicalDamageDealtToChampions => 27
```



```
physicalDamageTaken => 19
magicDamageTaken   => 23
totalDamageDealt    => 40
totalDamageDealtToChampions => 38
totalDamageTaken    => 30
totalHeal           => 16
kda                 => 9
kills               => 28
deaths              => 27
killingSprees       => 26
largestKillingSpree => 22
largestMultiKill    => 23
multikills          => 22
doubleKills         => 23
tripleKills         => 18
quadraKills         => 11
pentaKills          => 5
killsNearEnemyTurret => 19
maxKillDeficit      => 6
timePlayed          => 36
gameDuration        => 36
totalMinionsKilled  => 29
goldEarned          => 44
goldSpent           => 42
itemsPurchased      => 31
item0               => 1
item1               => 1
item2               => 1
item3               => 1
item4               => 1
item5               => 1
item6               => 1
spell1Casts         => 16
spell2Casts         => 15
spell3Casts         => 15
spell4Casts         => 4
turretKills         => 7
turretTakedowns     => 13
turretsLost         => 15
inhibitorsLost      => 17
gameEndedInEarlySurrender => 10
gameEndedInSurrender => 6
win                 => 9
```

2 Summoner Level Distribution

```
[26]: collection = db["match_info"]

# Initialize a Counter for summonerLevel
summoner_level_counts = Counter()

# Query all documents in the collection
for document in collection.find():
    # Extract participants from each document
    participants = document.get("info", {}).get("participants", [])
    for participant in participants:
        summoner_level = participant.get("summonerLevel")
        if summoner_level is not None:
            summoner_level_counts[summoner_level] += 1

# Convert the Counter to a DataFrame
summoner_level_df = pd.DataFrame(
    summoner_level_counts.items(), columns=["summonerLevel", "Count"]
).sort_values(by="summonerLevel")
print(summoner_level_df)
```

	Summoner_Level	Count
927	3	15
926	4	17
924	5	28
554	6	31
774	7	31
...
815	1558	1
1137	1570	1
1127	1672	1
1029	2158	1
971	3575	1

[1143 rows x 2 columns]

```
[46]: # Display the DataFrame
df = pd.DataFrame(summoner_level_df)
df.rename(columns={"Summoner_Level": "summonerLevel", "Count": "userCount"},
          inplace=True)

print(summoner_level_df)

max_summoner_level = df['summonerLevel'].max()
print(f"The maximum summoner level is: {max_summoner_level}")

max_user_count = df['userCount'].max()
```

```
print(f"The maximum user count is: {max_user_count}")
```

```
ticksX = np.arange(0, max_summoner_level, 100)
```

```
ticksY = np.arange(0, max_user_count, 100)
```

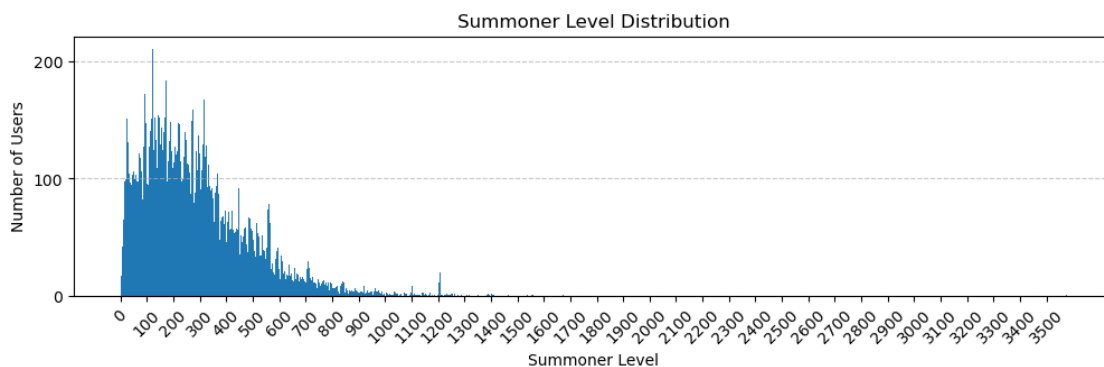
	Summoner_Level	Count
927	3	15
926	4	17
924	5	28
554	6	31
774	7	31
...
815	1558	1
1137	1570	1
1127	1672	1
1029	2158	1
971	3575	1

```
[1143 rows x 2 columns]
```

```
The maximum summoner level is: 3575
```

```
The maximum user count is: 211
```

```
[47]: plt.figure(figsize=(12, 3))
plt.bar(df['summonerLevel'], df['userCount'], width=2.4)
plt.xlabel('Summoner Level')
plt.ylabel('Number of Users')
plt.title('Summoner Level Distribution')
plt.xticks(ticksX, rotation=45)
plt.yticks(ticksY)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



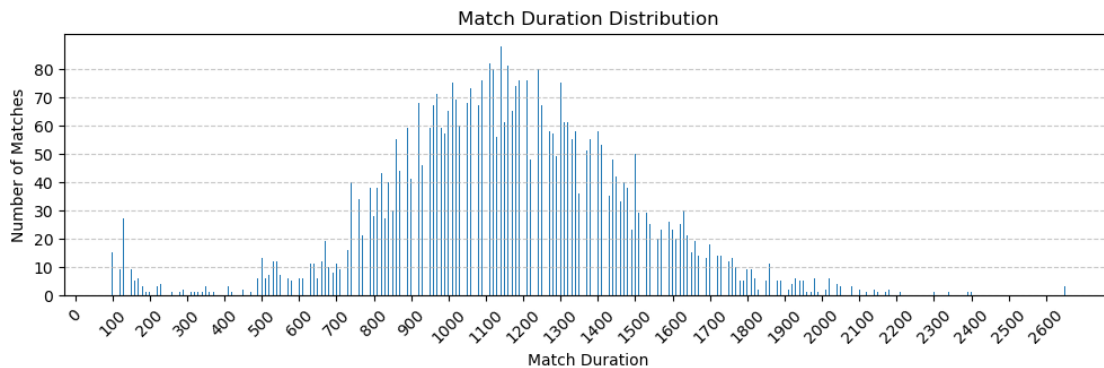
3 Match Duration Distribution

```
[25]: df = pd.DataFrame(results)
df.rename(columns={"_id": "matchDuration", "count": "durationCount"},
          inplace=True)

maxX = max(results, key=lambda x: x['_id'])['_id']
maxY = max(results, key=lambda x: x['count'])['count']

ticksX = np.arange(0, maxX+10, 100)
ticksY = np.arange(0, maxY+0.5, 10)

plt.figure(figsize=(12, 3))
plt.bar(df['matchDuration'], df['durationCount'], width=2.4)
plt.xlabel('Match Duration')
plt.ylabel('Number of Matches')
plt.title('Match Duration Distribution')
plt.xticks(ticksX, rotation=45)
plt.yticks(ticksY)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



```
[ ]:
```