



# Prediksi & Analisis : Valorant Pro Matches

Kevin Alexander (2106705026)  
Emir Shamsuddin Fadhlurrahman (2106632541)  
Michael Christlambert Sinanta (2106750414)  
Vinsen Wijaya (2106637776)

# Daftar Isi

-  **Latar Belakang dan Tujuan**
-  **Dataset**
-  **Exploratory Data Analysis**
  - Distribusi Agent yang Digunakan oleh Pemain di Setiap Patch
  - Persebaran Nilai ACS pada Setiap Agent
  - Komposisi Agent yang Paling Banyak Memenangkan Pertandingan
  - Tim yang paling banyak memenangkan Stage Grand Final
  - Eksplorasi Mandiri 1:** Rata-Rata Kill/Death setiap Agent per Map
  - Eksplorasi Mandiri 2:** Distribusi Map yang Digunakan oleh Pemain di Setiap Patch
-  **Classification**
-  **Regression**
-  **Clustering**
  - Hasil Clustering
  - Analisis Hasil Clustering
-  **Referensi**

# Latar Belakang & Tujuan

Valorant adalah sebuah permainan video dalam genre First-Person Shooter (FPS) yang dikembangkan dan diterbitkan oleh Riot Games. Permainan ini pertama kali diumumkan dengan nama kode "Project A" pada Oktober 2019, dan diluncurkan pada 2 Juni 2020. Sejak rilisnya, "Valorant" telah menjadi salah satu game FPS yang paling populer, terutama di kalangan kompetitif dan esports.

---

Tujuan dari proyek ini adalah :

- Melakukan eksplorasi data yang didapat dari dataset matches, agent, games, dan agent\_encode dan mendapatkan insight untuk menjawab pertanyaan EDA.
- Membangun model klasifikasi untuk memprediksi agen yang akan digunakan oleh setiap pemain di setiap babak.
- Membangun model untuk memprediksi nilai rata-rata ACS dari masing-masing tim untuk sebuah pertandingan.
- Melakukan clustering pada dataset serta memberikan analisis dan insight yang didapat dari cluster yang telah dibuat.

# Dataset



# Informasi Dataset Matches 12 kolom x 6401 baris

Berisi informasi tentang tim yang berpartisipasi dalam sebuah turnamen. Pertandingan umumnya diadakan dengan aturan "best of 3," di mana tim yang memenangkan babak terbanyak akan menjadi pemenang pertandingan.

Nama	Tipe	Description
MatchId	int64	Id pertandingan
Date	object	Tanggal pertandingan
Patch	object	Versi Valorant saat pertandingan dilakukan
EventID	int64	Id turnamen
EventName	object	Nama turnamen
EventStage	object	Tahap turnamen

Nama	Tipe	Description
Team1ID	int64	Id tim 1
Team2ID	int64	Id tim 2
Team1	int64	Nama tim 1
Team2	int64	Nama tim 2
Team1_MapScore	object	Banyak babak yang dimenangkan tim 1
Team2_MapScore	int64	Banyak babak yang dimenangkan tim 2

# Informasi Dataset Games 18 kolom x 12961 baris

Mendokumentasikan detail babak yang dimainkan oleh kedua tim yang bersaing dalam turnamen.

Nama	Tipe	Description
GameID	int64	Id babak
MatchID	int64	Id pertandingan
Map	object	Peta yang digunakan
Team1ID	int64	Id tim 1
Team2ID	int64	Id tim 2
Team1	object	Nama tim 1
Team2	object	Nama tim 2
Winner	int64	Tim yang menjadi pemenang

Nama	Tipe	Description
Team1_Eco	float64	Banyak ronde eco yang dilakukan tim 1
Team1_SemiEco	float64	Banyak ronde semi eco yang dilakukan tim 1
Team1_SemiBuy	float64	Banyak ronde semi buy yang dilakukan tim 1
Team1_FullBuy	float64	Banyak ronde full buy yang dilakukan tim 1
Team1_TotalRounds	int64	Banyak ronde yang dimenangkan tim 1
Team2_Eco	float64	Banyak ronde eco yang dilakukan tim 2
Team2_SemiEco	float64	Banyak ronde semi eco yang dilakukan tim 2
Team2_SemiBuy	float64	Banyak ronde semi buy yang dilakukan tim 2
Team2_FullBuy	float64	Banyak ronde full buy yang dilakukan tim 2
Team2_TotalRounds	int64	Banyak ronde yang dimenangkan tim 2

# Informasi Dataset Scores 28 kolom x 128779 baris

Merupakan catatan akhir dari status para pemain profesional yang bertanding dalam babak tersebut

Nama	Tipe	Description
GameID	int64	Id babak
PlayerID	float64	Id pemain
PlayerName	object	Username pemain
TeamAbbreviation	object	Singkatan nama tim
Agent	object	Agent/karakter yang digunakan
ACS	float64	Skor rata-rata dari pemain
Kills	float64	Jumlah musuh yang dikalahkan
Deaths	float64	Total pemain berhasil dieliminasi
Assists	float64	Jumlah kontribusi pemain dalam membantu rekannya untuk mengalahkan musuh
PlusMinus	float64	Selisih antara jumlah mengalahkan musuh (Kills) dan jumlah tereliminasi (Deaths)
KAST_Percent	float64	Besar kontribusi pemain dalam pertandingan

Nama	Tipe	Description
ADR	float64	Rata-rata serangan yang diberikan oleh pemain ke pihak musuh
HS_Percent	float64	Persentase serangan yang diberikan mengenai kepala musuh
FirstKills	float64	Total pemain berhasil melakukan eliminasi pertama kali pada suatu ronde
FirstDeaths	float64	Total pemain dieliminasi pertama kali pada suatu ronde
FKFD_PlusMinus	float64	Selisih antara jumlah First Kills dan First Deaths
Num_2Ks, Num_3ks, Num_4ks, Num_5ks	float64	Jumlah pemain berhasil mengeliminasi X musuh dalam suatu ronde
OnevOne, OnevTwo, OnevThree, OnevFour, OnevFive	float64	Jumlah pemain berhasil mengalahkan X musuh pada suatu timeframe
Econ	float64	Besar serangan yang dihasilkan untuk setiap 1000 creds (mata uang pada permainan) yang digunakan untuk membeli perlengkapan
Plants	float64	Jumlah pemain berhasil menanam spike (bom) pada suatu ronde
Defuses	float64	Jumlah pemain berhasil menjinakkan spike (bom) pada suatu ronde

# Preprocessing



# Dataset Scores

## Data Cleaning

- Menghilangkan kolom KAST Percent dan kolom No
- Melakukan drop untuk menghilangkan baris-baris yang tidak memiliki informasi FirstKills, ACS, Assists, Deaths, Kills, Agent, Defused, PlayerID, dan TeamAbbreviation

	Total	Percent
KAST_Percent	126142	97.952306
Defuses	10195	7.916663
OnevThree	10195	7.916663
Num_3Ks	10195	7.916663
Num_2Ks	10195	7.916663
OnevOne	10195	7.916663
Num_5Ks	10195	7.916663
OnevTwo	10195	7.916663
Num_4Ks	10195	7.916663
OnevFour	10195	7.916663
OnevFive	10195	7.916663
Econ	10195	7.916663
Plants	10195	7.916663
HS_Percent	9472	7.355236
FirstDeaths	9465	7.349801
FKFD_PlusMinus	9465	7.349801
ADR	8875	6.891652
Agent	4100	3.183749
TeamAbbreviation	2016	1.565473
PlusMinus	1753	1.361247
FirstKills	530	0.411558
ACS	530	0.411558
Assists	490	0.380497
Deaths	490	0.380497
Kills	490	0.380497
PlayerID	87	0.067558

Informasi missing value dari scores.csv

## Data Imputation

HS\_Percent 7 0.005988

Informasi missing value HS\_Percent

Melakukan pengisian missing value pada kolom HS\_Percent menggunakan nilai mean.

## Memastikan setiap *game* memiliki 10 *player*

Valorant merupakan game yang dimainkan oleh 10 *player* dalam waktu bersamaan sehingga kami memastikan setiap game memiliki informasi 10 *player* yang terlibat dalam game tersebut.

# Dataset Games

## Data Cleaning

- Menghilangkan kolom No
- Melakukan drop untuk menghilangkan baris-baris yang memiliki *missing value*, termasuk baris dengan nilai kolom Map berupa ‘TBD’

	Total	Percent
Team1_SemiEco	1034	7.977779
Team1_SemiBuy	1034	7.977779
Team2_FullBuy	1034	7.977779
Team2_SemiBuy	1034	7.977779
Team2_SemiEco	1034	7.977779
Team2_Eco	1034	7.977779
Team1_FullBuy	1034	7.977779
Team1_Eco	1034	7.977779

Informasi missing value dari games.csv

# Dataset Matches

## Data Cleaning

Menghilangkan kolom No

## Data Imputation

*Missing Value* pada kolom Patch diisi secara manual dengan menggunakan informasi kolom Date dan informasi tentang Patch dari Valorant Wiki.

```
from datetime import datetime

start_patch0 = datetime(2020, 4, 15)
end_patch0 = datetime(2020, 6, 2)

start_patch1 = datetime(2020, 6, 2)
end_patch1 = datetime(2021, 1, 12)

start_patch2 = datetime(2021, 1, 12)
end_patch2 = datetime(2021, 6, 22)

start_patch3 = datetime(2021, 6, 22)
end_patch3 = datetime(2022, 1, 11)

def handle_patch_null(row):
    if pd.isnull(row['Patch']):
        date = row['Date'][:10]
        date_object = datetime.strptime(date, '%Y-%m-%d')
        if start_patch0 <= date_object < end_patch0:
            row['Patch'] = 'Patch 0.0'
        elif start_patch1 <= date_object < end_patch1:
            row['Patch'] = 'Patch 1.0'
        elif start_patch2 <= date_object < end_patch2:
            row['Patch'] = 'Patch 2.0'
        elif start_patch3 <= date_object < end_patch3:
            row['Patch'] = 'Patch 3.0'
    return row
```

snippet kode untuk Data Imputation

# Dataset Gabungan

Ketiga dataset digabung menjadi satu

- Dataset Matches dan Games digabung menggunakan atribut MatchID
- Hasil penggabungan tadi digabung dengan dataset Scores menggunakan GameID

```
Index(['GameID', 'PlayerID', 'PlayerName', 'TeamAbbreviation', 'Agent', 'ACS',
       'Kills', 'Deaths', 'Assists', 'PlusMinus', 'ADR', 'HS_Percent',
       'FirstKills', 'FirstDeaths', 'FKFD_PlusMinus', 'Num_2Ks', 'Num_3Ks',
       'Num_4Ks', 'Num_5Ks', 'OnevOne', 'OnevTwo', 'OnevThree', 'OnevFour',
       'OnevFive', 'Econ', 'Plants', 'Defuses', 'MatchID', 'Map', 'Team1ID_x',
       'Team2ID_x', 'Team1_x', 'Team2_x', 'Winner', 'Team1_Eco',
       'Team1_SemiEco', 'Team1_SemiBuy', 'Team1_FullBuy', 'Team1_TotalRounds',
       'Team2_Eco', 'Team2_SemiEco', 'Team2_SemiBuy', 'Team2_FullBuy',
       'Team2_TotalRounds', 'Date', 'Patch', 'EventID', 'EventName',
       'EventStage', 'Team1ID_y', 'Team2ID_y', 'Team1_y', 'Team2_y',
       'Team1_MapScore', 'Team2_MapScore'],
      dtype='object')
```

Kolom-kolom pada dataframe setelah digabungkan



## x dan y

Hasil penggabungan memecah beberapa fitur terkait informasi Tim menjadi x dan y. Kami mengambil fitur yang ditandai oleh y karena memiliki informasi yang lebih lengkap ketimbang x.

# Dataset Final

51 kolom x 108250 baris

No	Nama	Tipe
0	GameID	int64
1	PlayerID	float64
2	PlayerName	object
3	TeamAbbreviation	object
4	Agent	object
5	ACS	float64
6	Kills	float64
7	Deaths	float64
8	Assists	float64
9	PlusMinus	float64
10	ADR	float64
11	HS_Percent	float64
12	FirstKills	float64
13	FirstDeaths	float64
14	FKFD_PlusMinus	float64
15	Num_2Ks	float64
16	Num_3K	float64

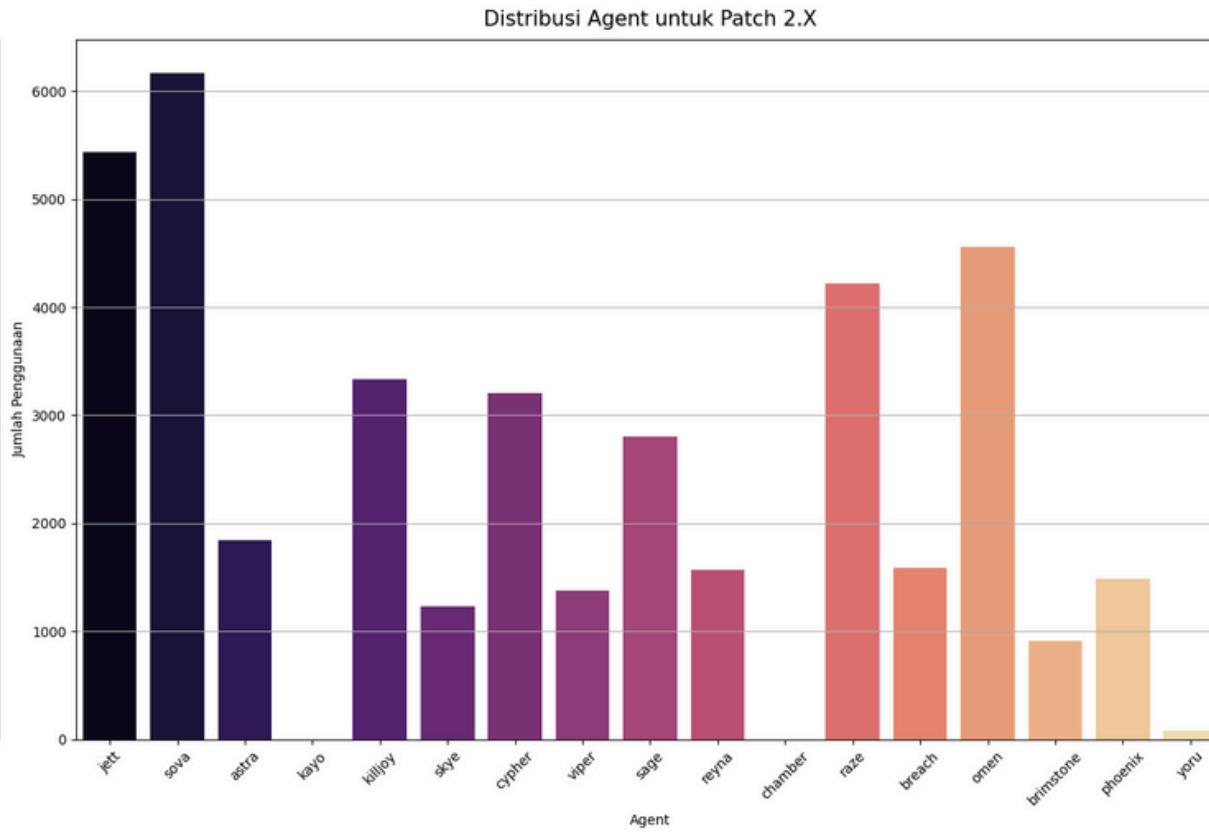
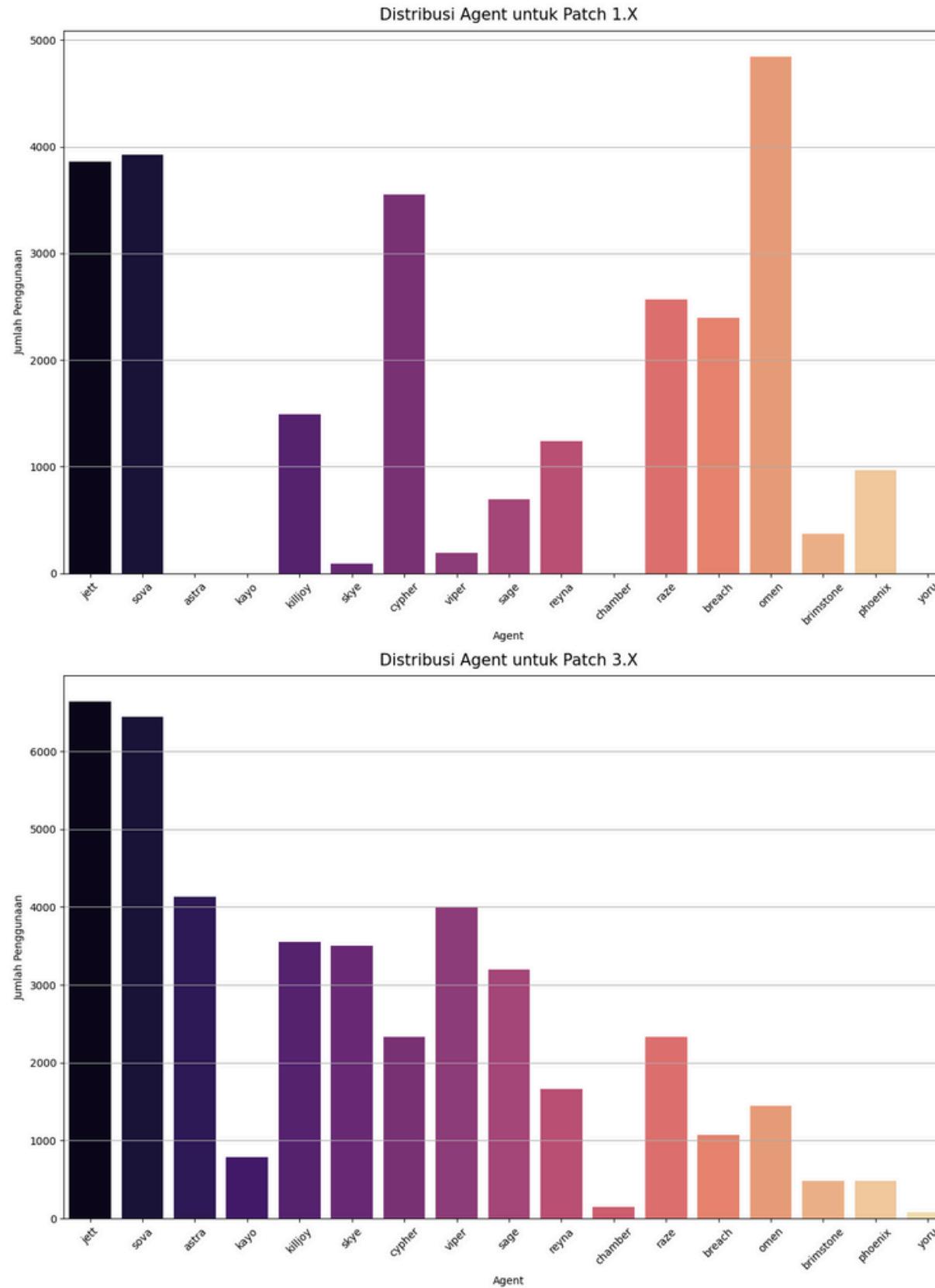
No	Nama	Tipe
17	Num_4K	float64
18	Num_5K	float64
19	OnevOne	float64
20	OnevTwo	float64
21	OnevThree	float64
22	OnevFour	float64
23	OnevFive	float64
24	Econ	float64
25	Plants	float64
26	Defuses	float64
27	MatchID	int64
28	Map	object
29	Winner	int64
30	Team1_Eco	float64
31	Team1_SemiEco	float64
32	Team1_SemiBuy	float64
33	Team1_FullBuy	float64

No	Nama	Tipe
34	Team1_TotalRounds	int64
35	Team2_Eco	float64
36	Team2_SemiEco	float64
37	Team2_SemiBuy	float64
38	Team2_FullBuy	float64
39	Team2_TotalRounds	int64
40	Date	object
41	Patch	float64
42	EventID	int64
43	EventName	object
44	EventStage	object
45	Team1ID	int64
46	Team2ID	int64
47	Team1	object
48	Team2	object
49	Team1_MapScore	int64
50	Team2_Mapscore	int64

# Exploratory Data Analysis



# Distribusi agent yang digunakan oleh pemain untuk setiap patch



Untuk Patch 1.X dan 2.X, Agent yang tidak memiliki jumlah penggunaan artinya belum muncul pada Patch tersebut.

Patch 1.X: Astra, Kayo, Chamber, Yoru

Patch 2.X: Kayo, Chamber

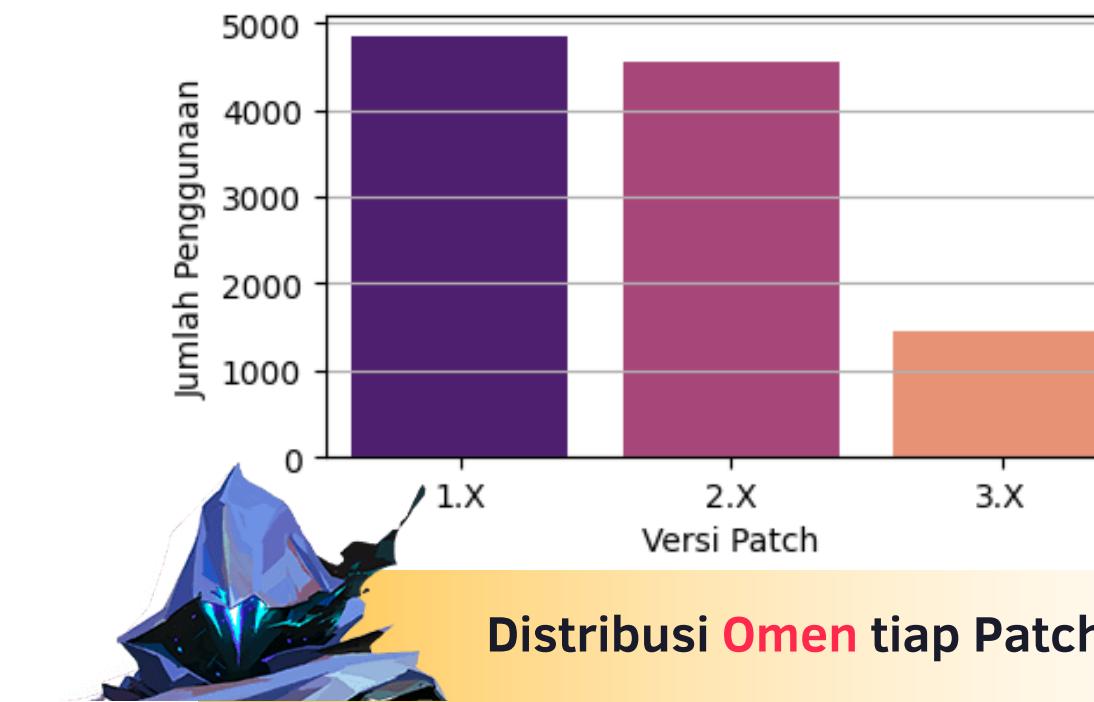
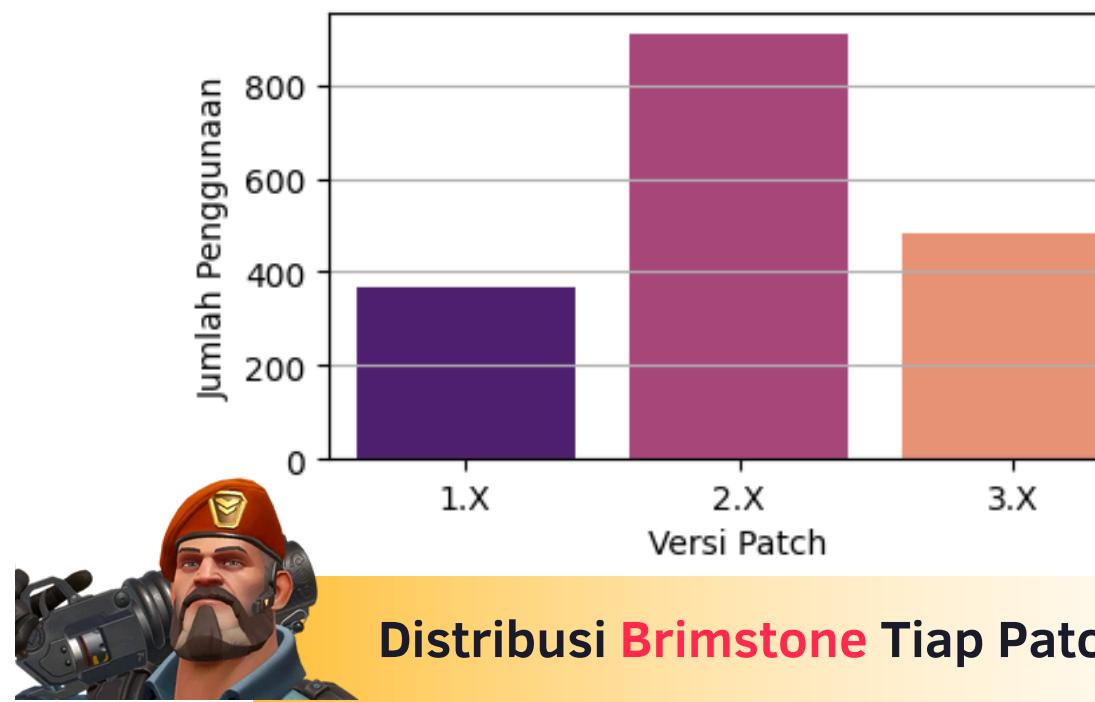
**Agent yang paling banyak digunakan oleh pemain di setiap patchnya :**

1.X = Omen

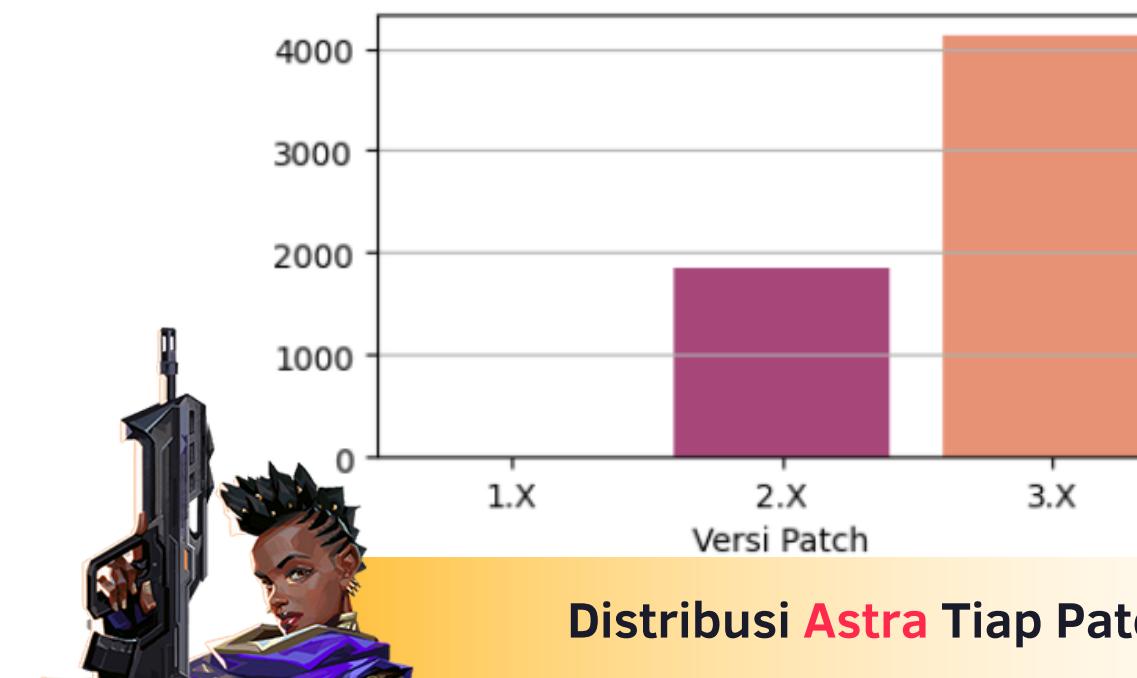
2.X = Sova

3.X = Jett

# Beberapa Agent yang jumlah penggunaannya menurun pada patch berbeda



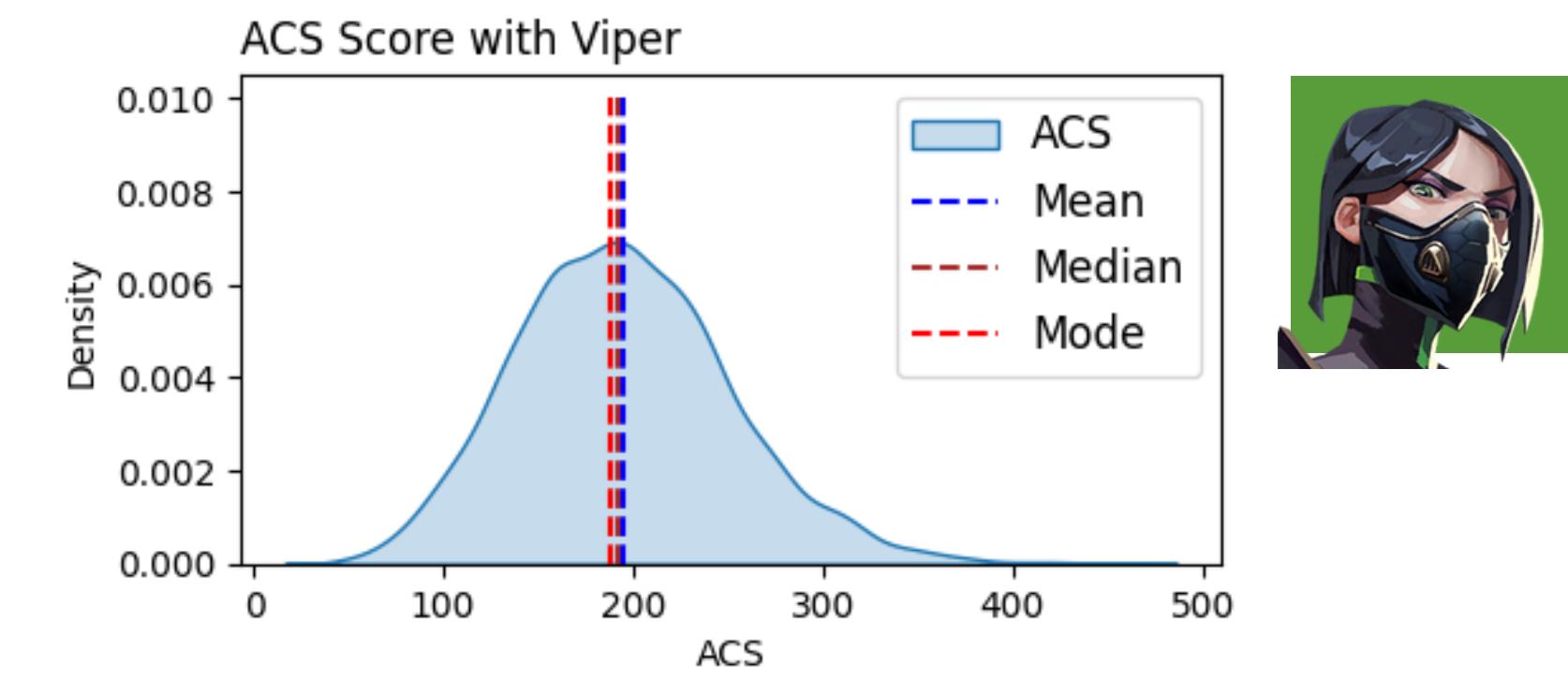
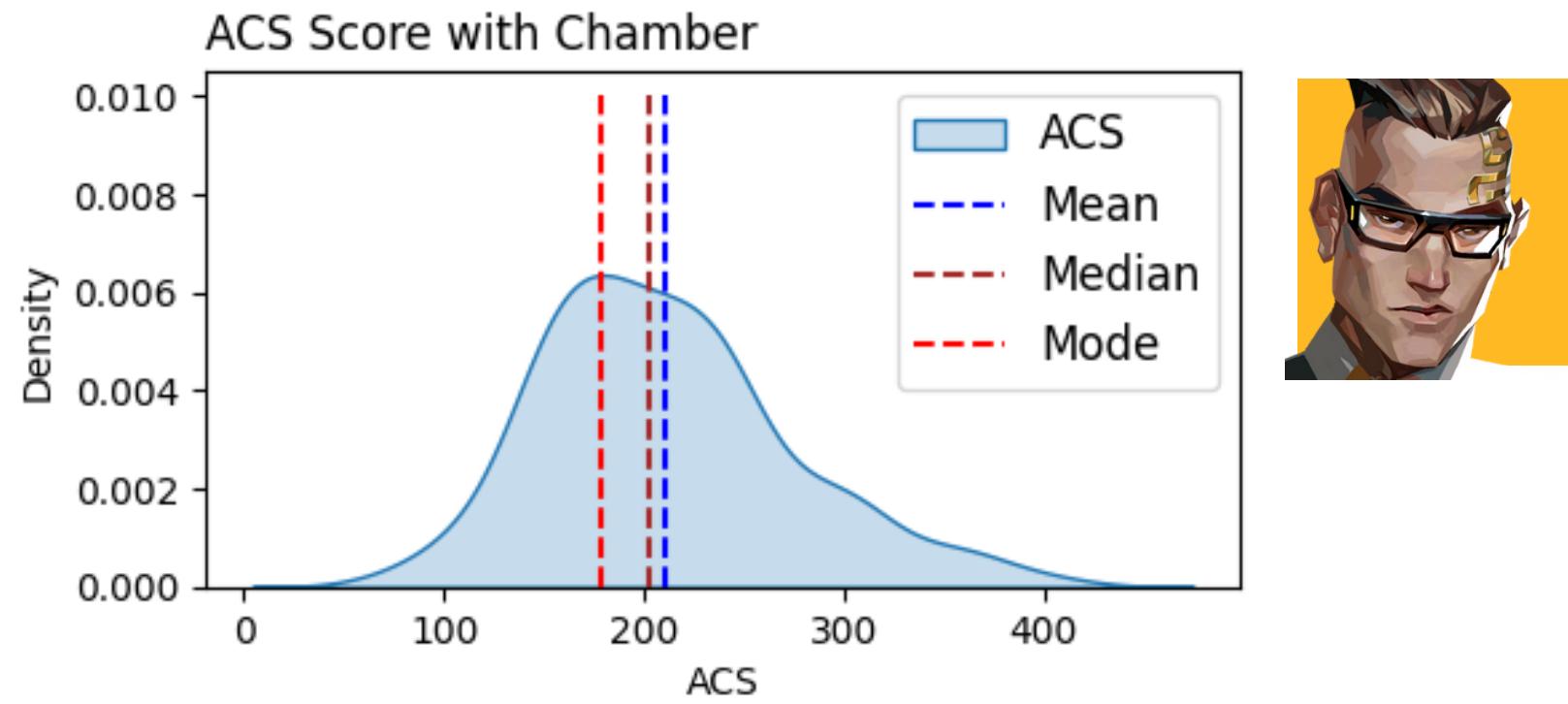
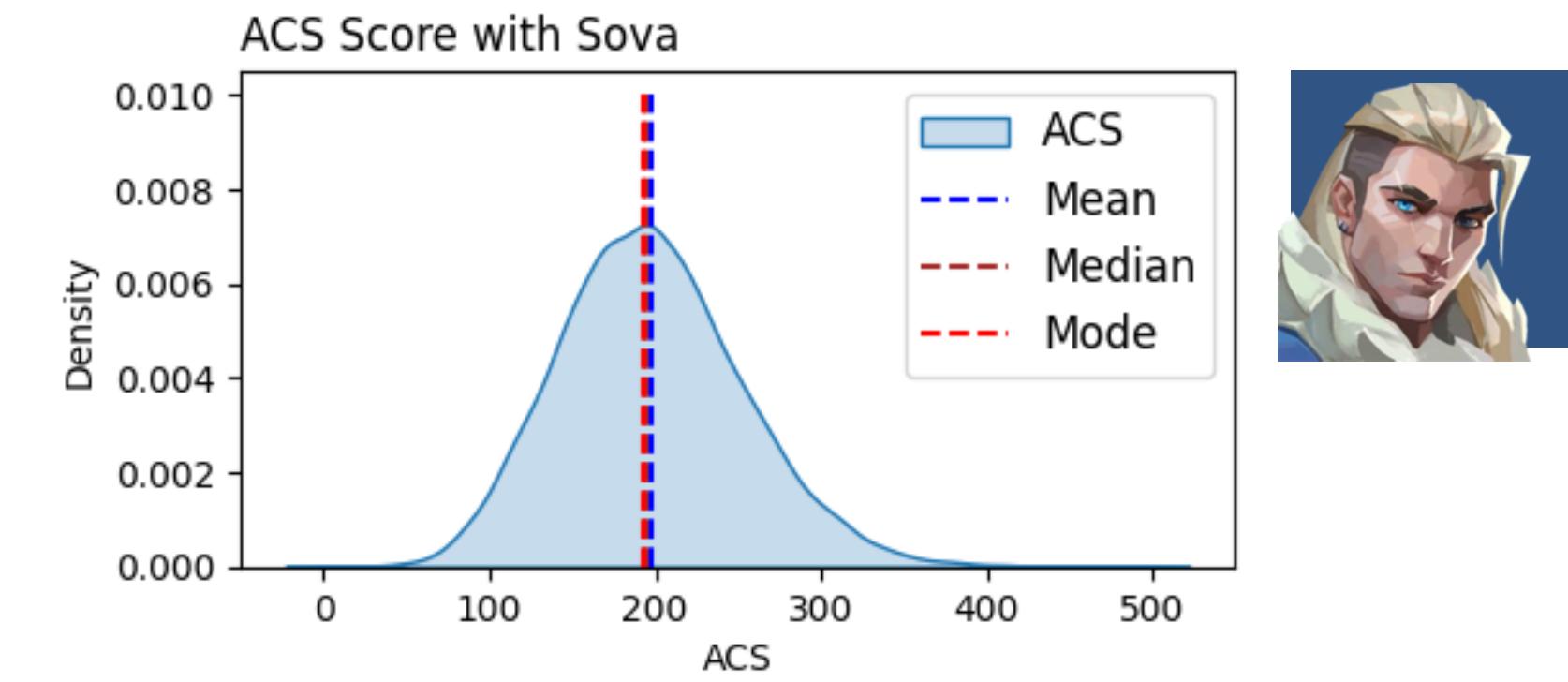
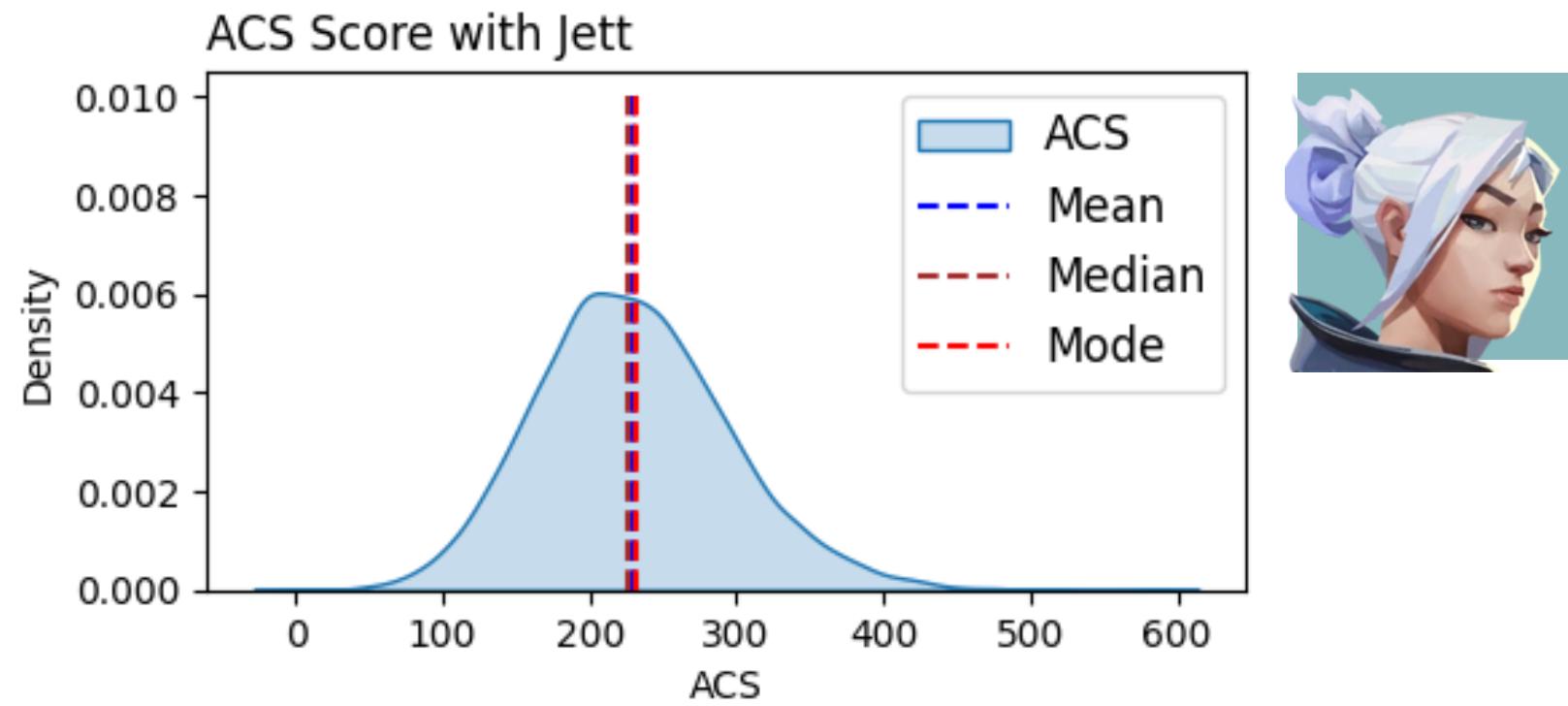
Penurunan jumlah penggunaan disebabkan oleh kemunculan Agent baru dengan peran yang sama.



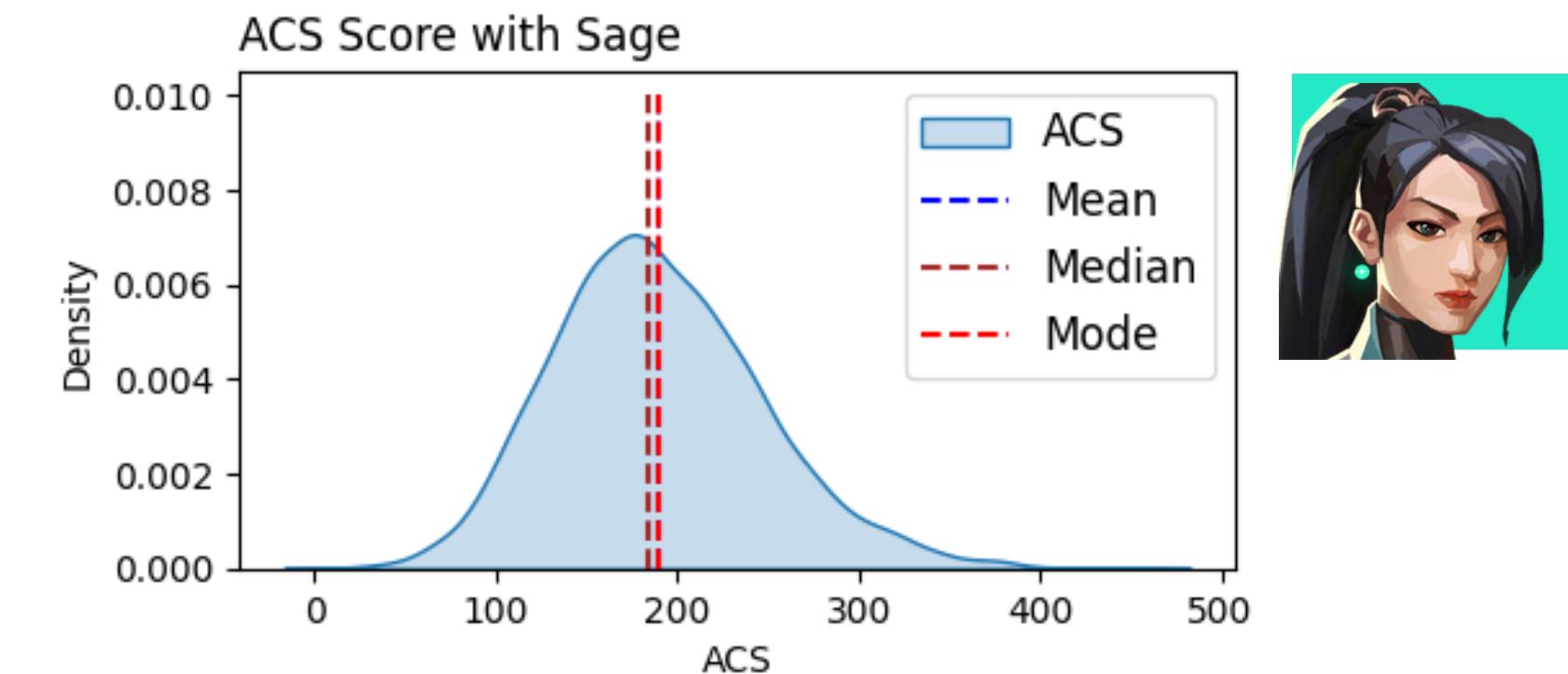
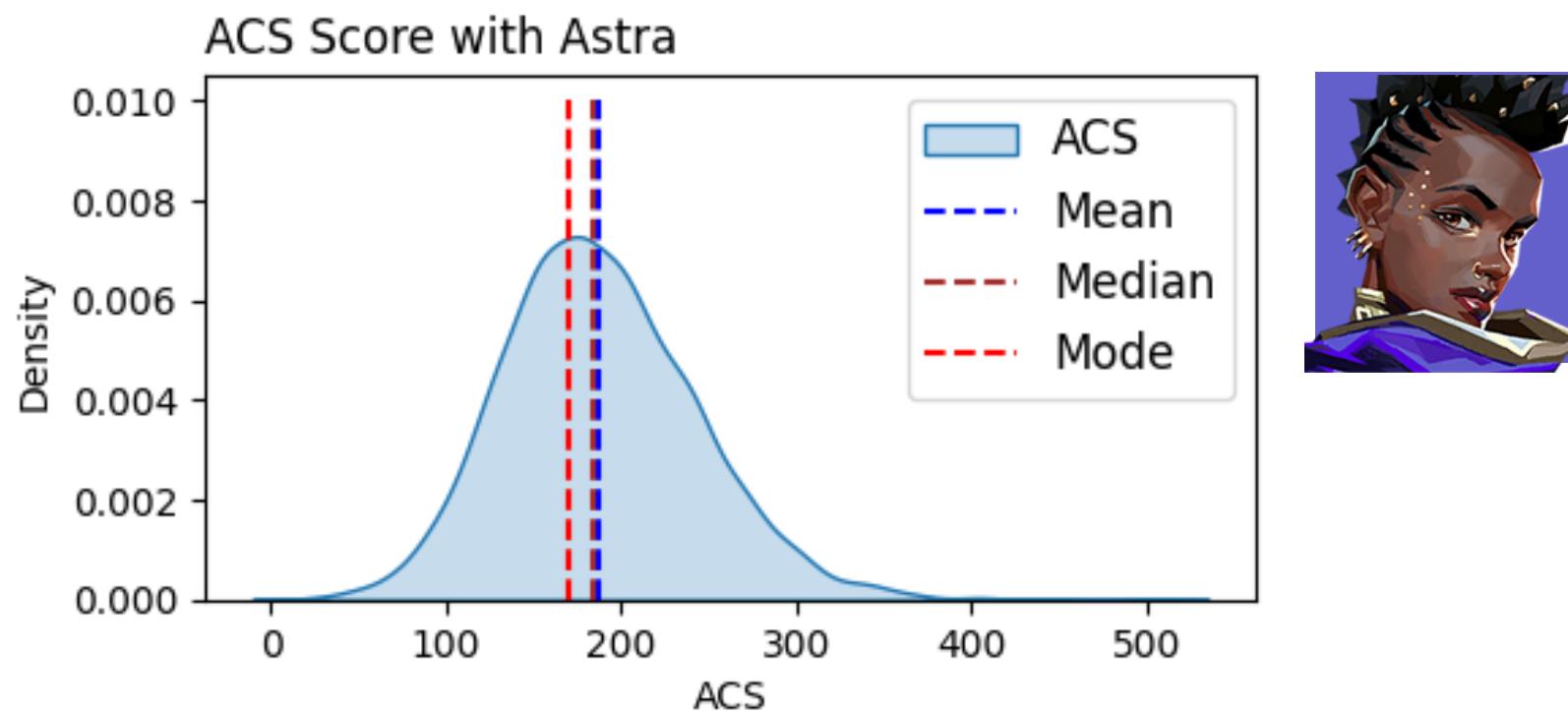
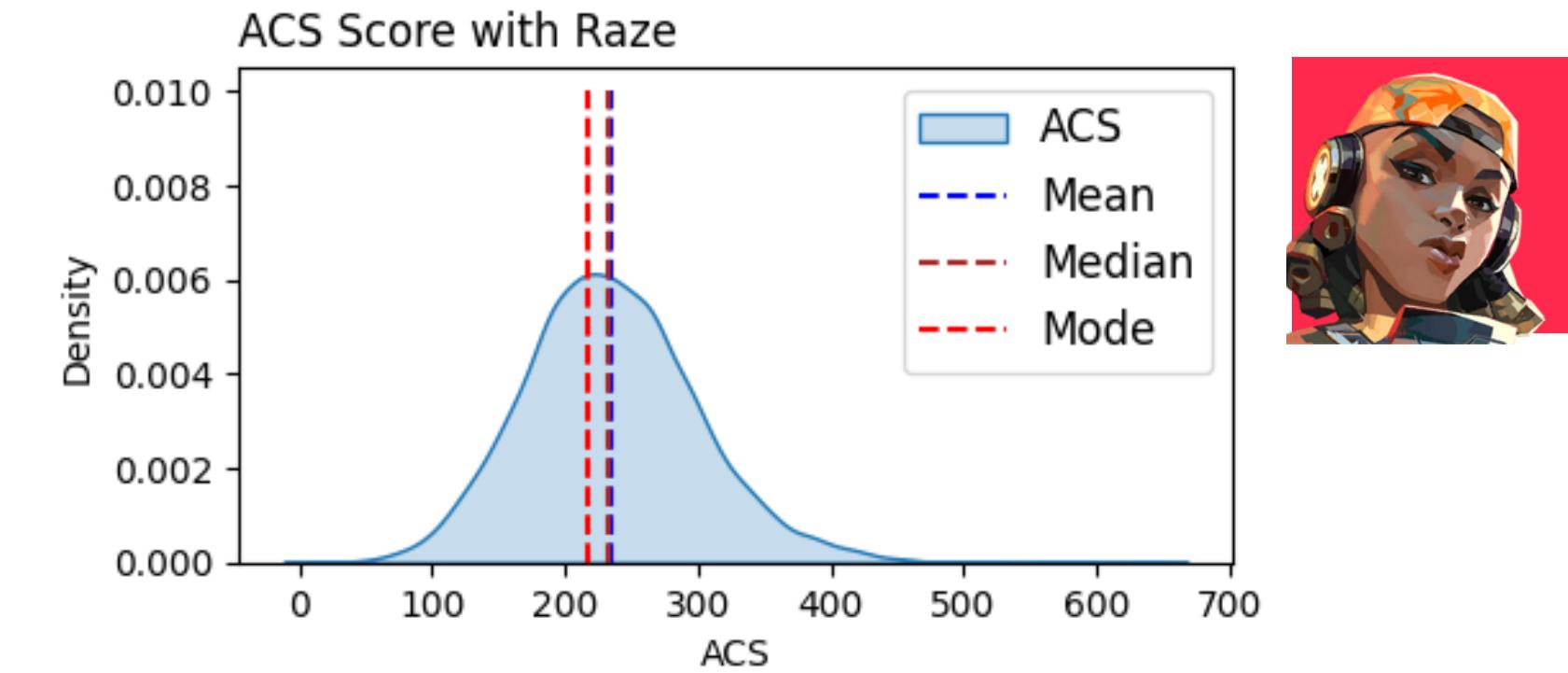
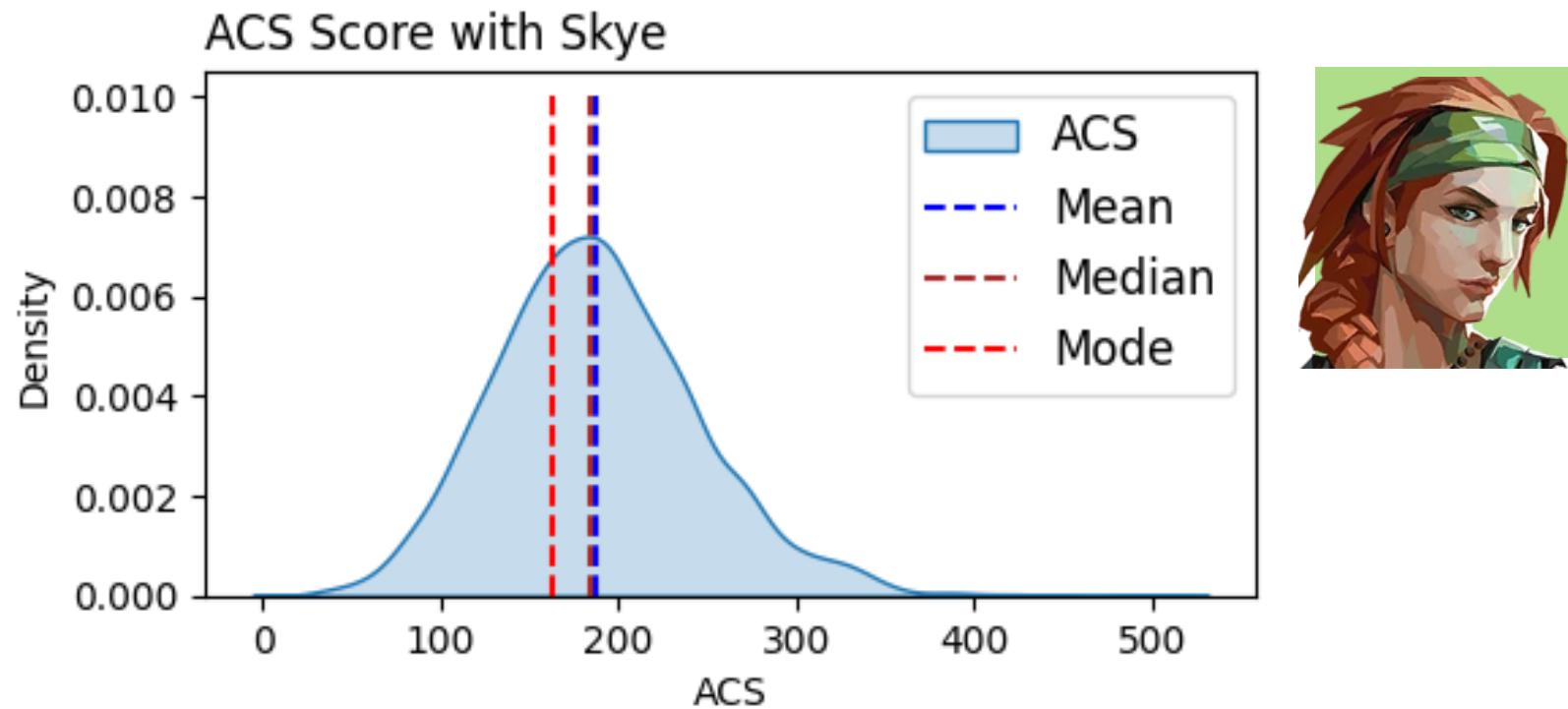
## Astra

- Baru muncul di Patch 2.04
- Mendominasi jumlah penggunaan untuk role Controller di Patch 3.X

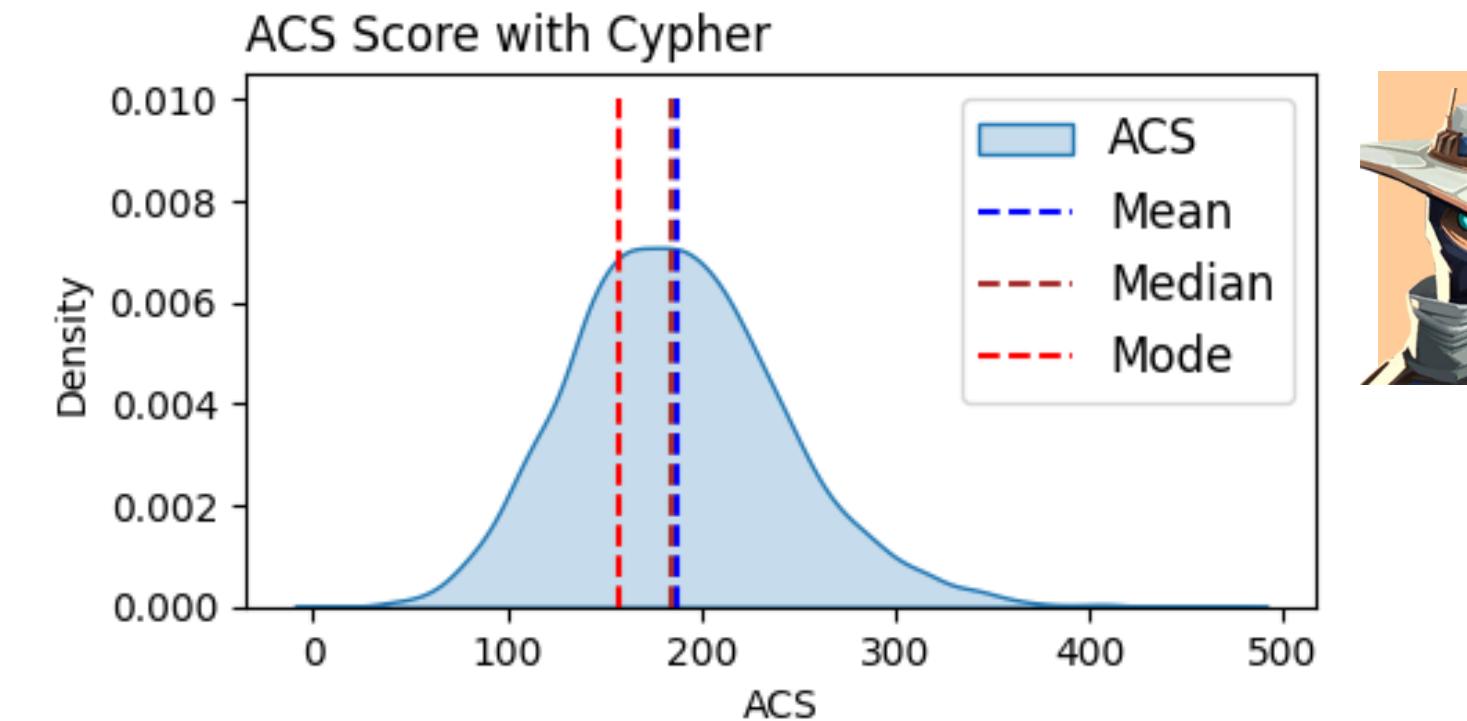
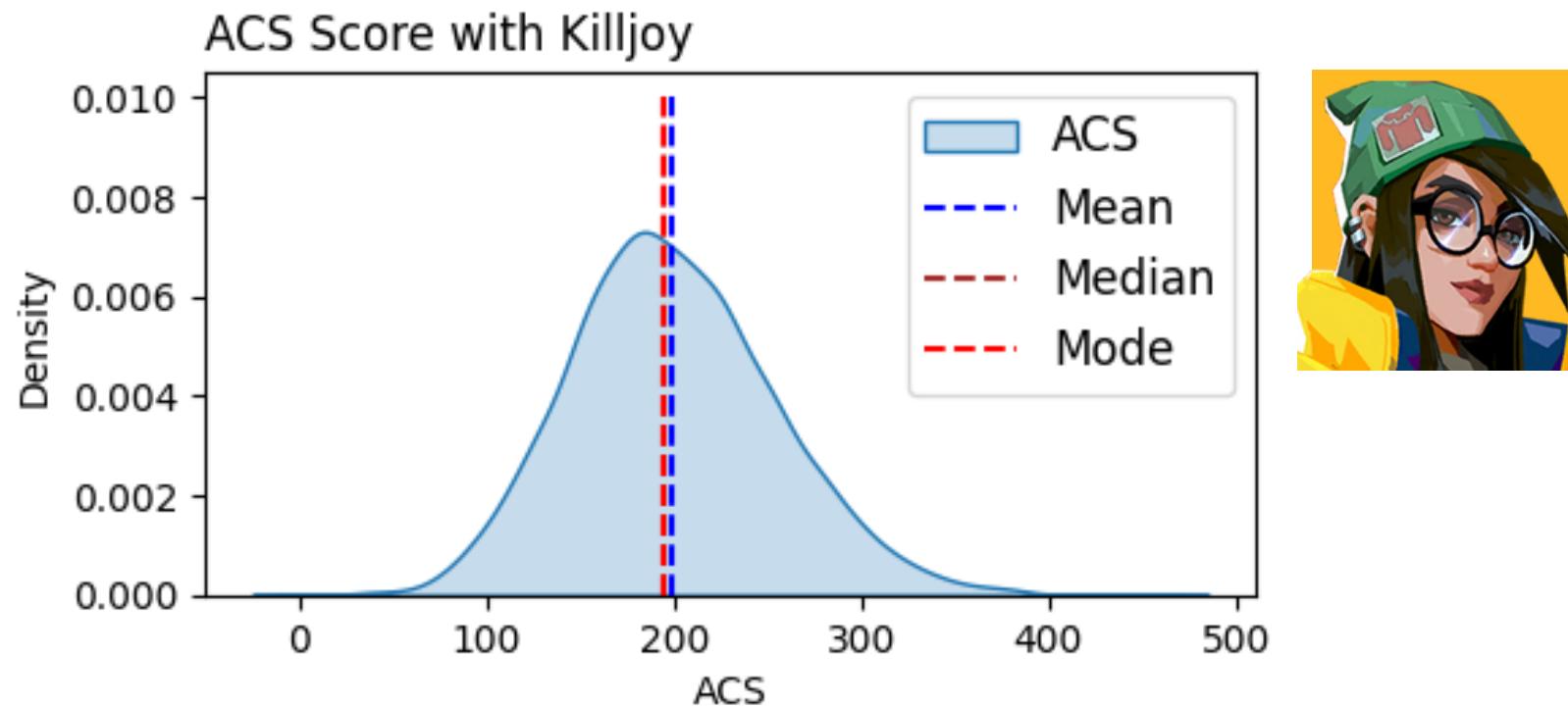
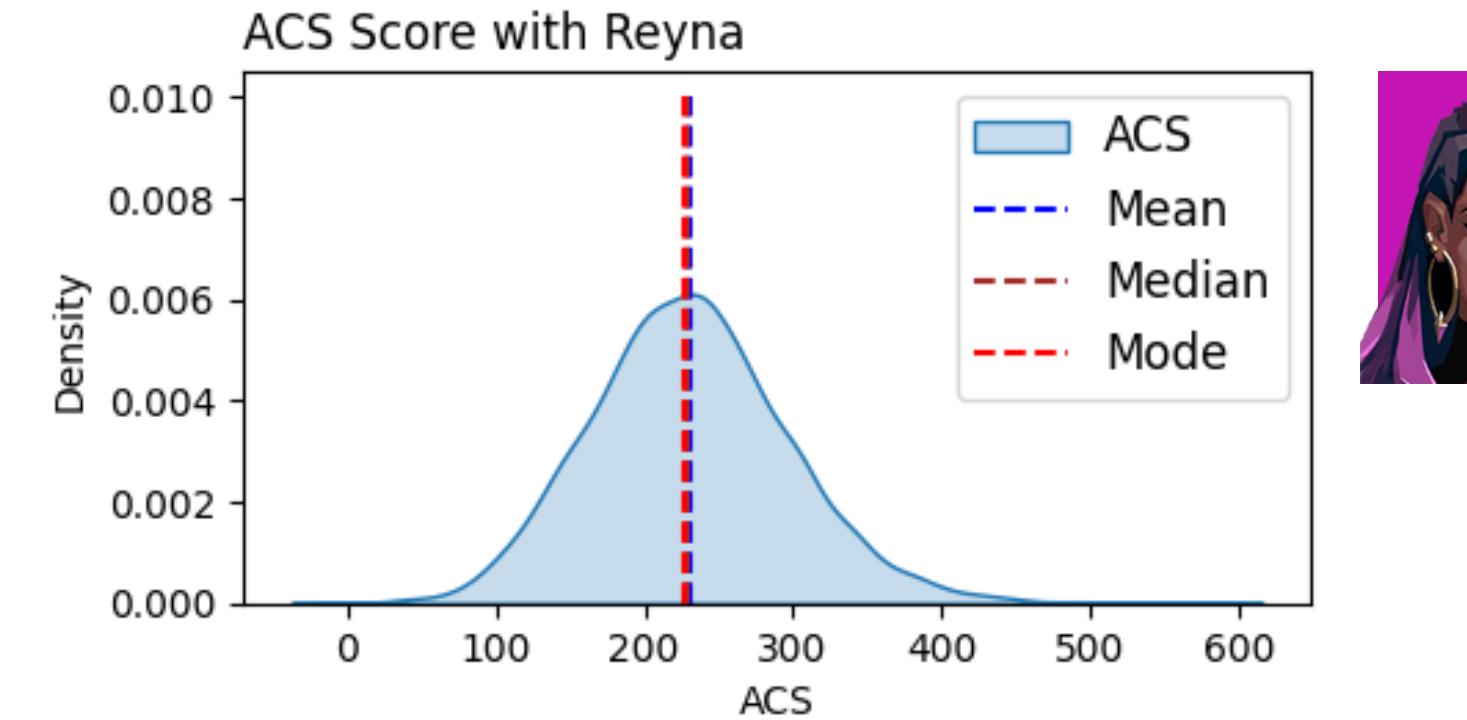
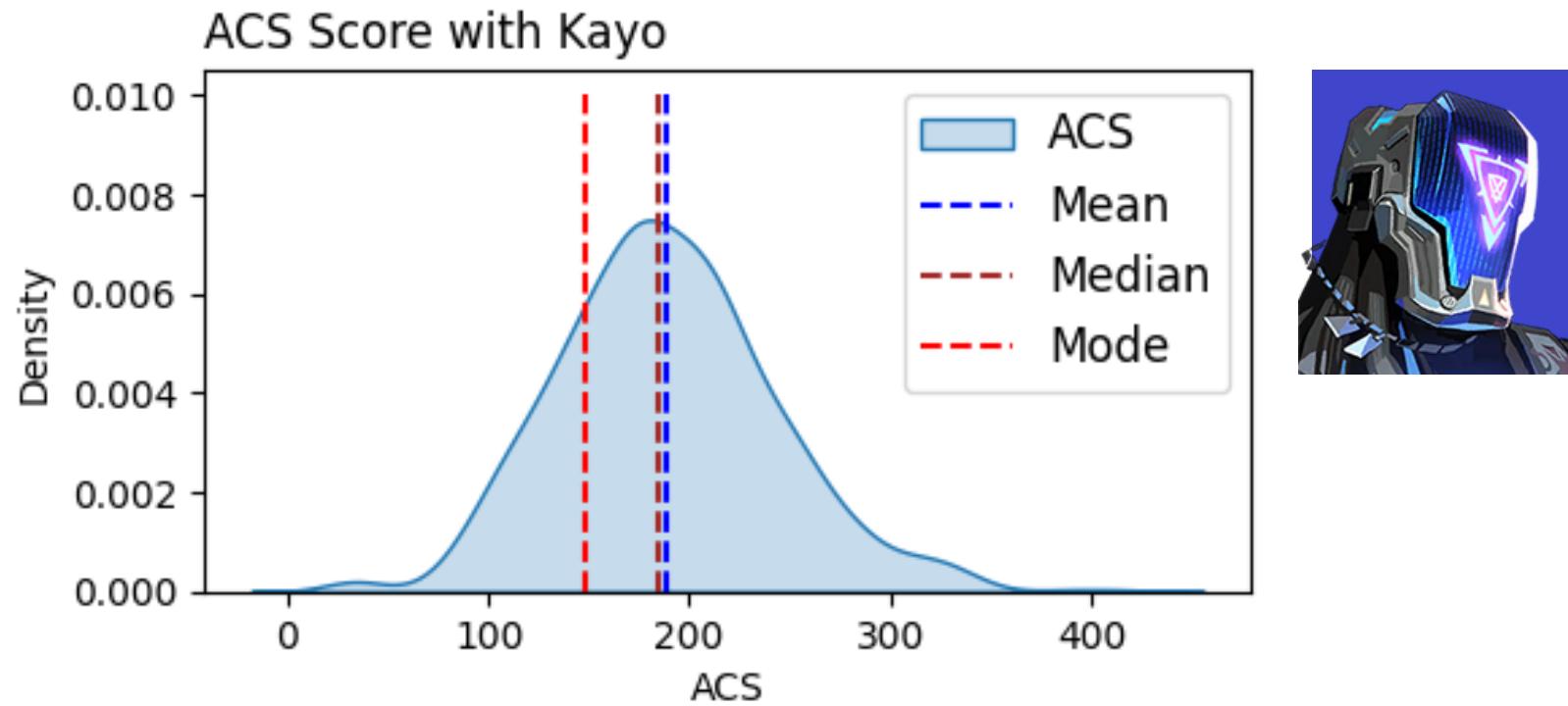
# Persebaran Nilai ACS pada setiap Agent (I)



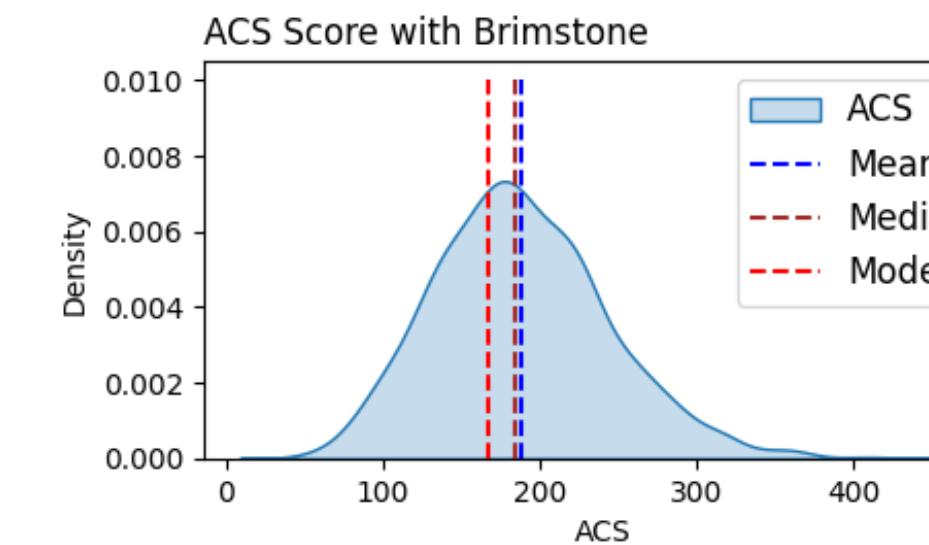
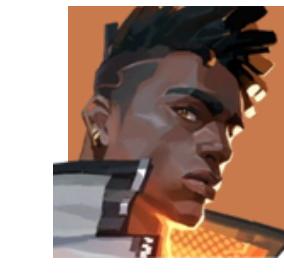
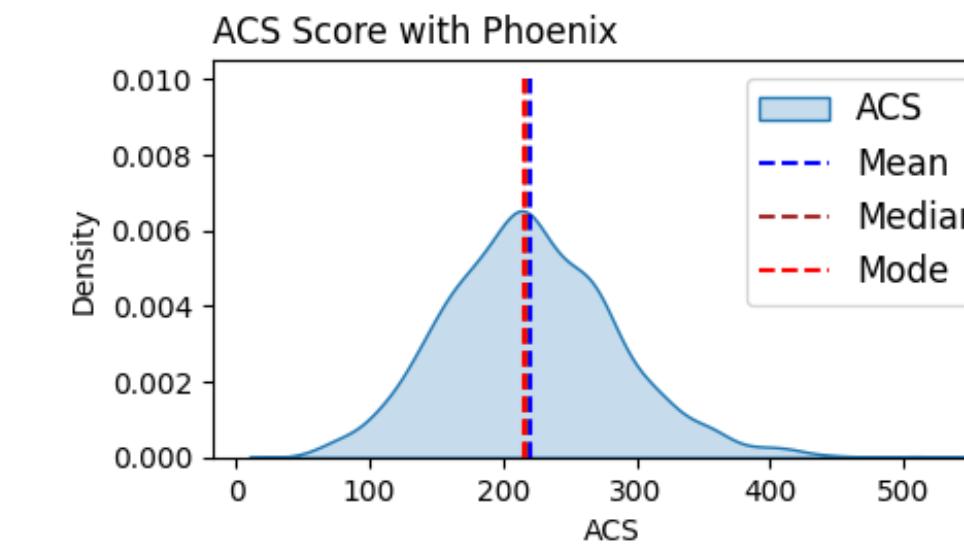
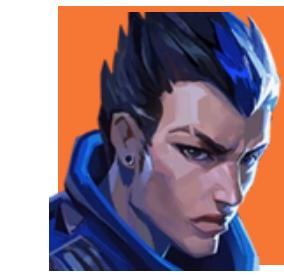
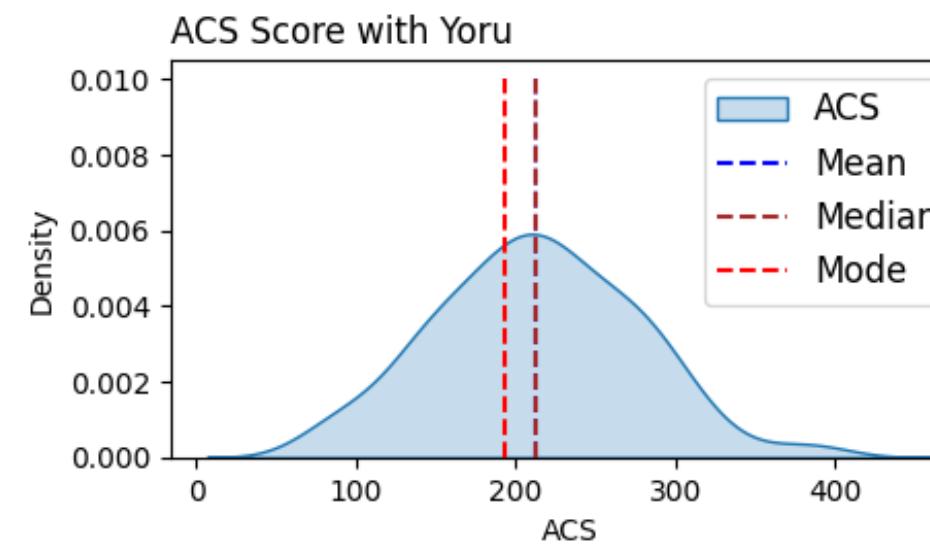
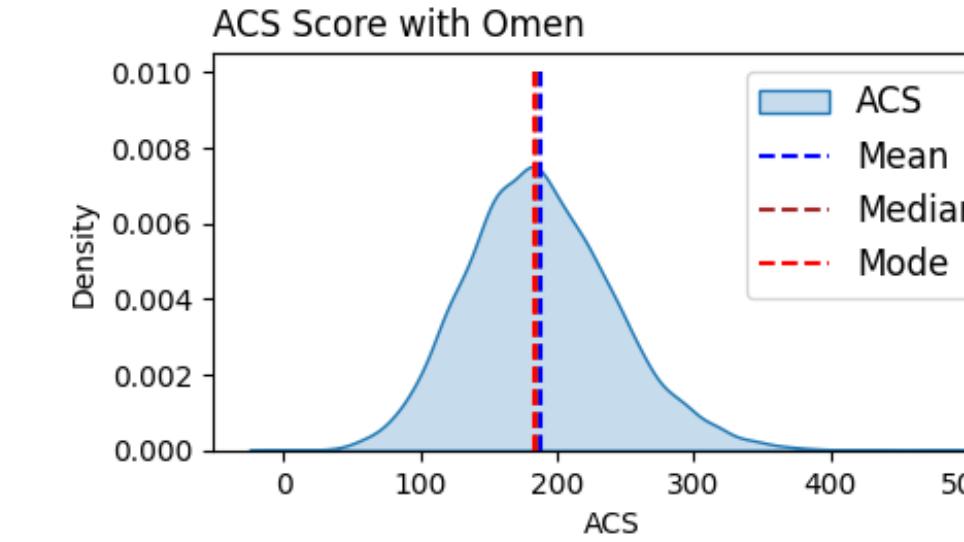
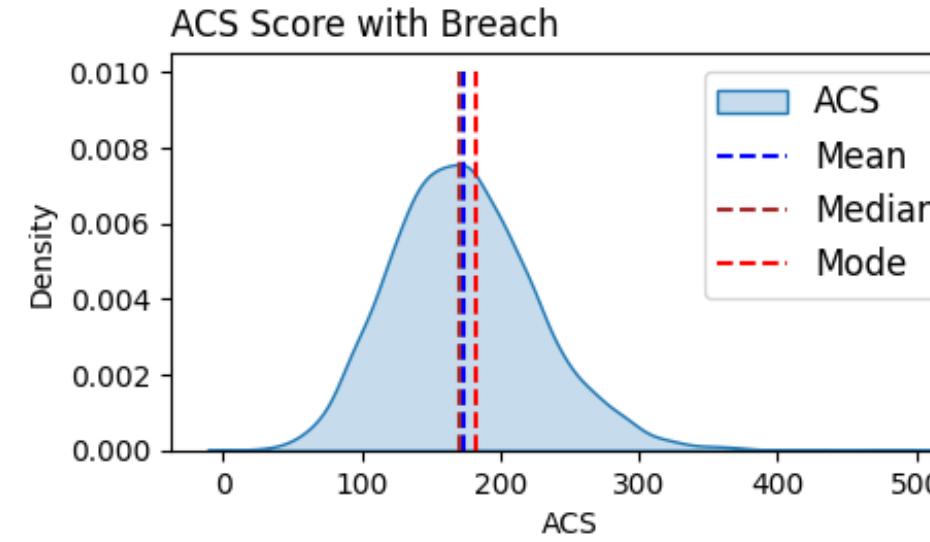
# Persebaran Nilai ACS pada setiap Agent (2)



# Persebaran Nilai ACS pada setiap Agent (3)

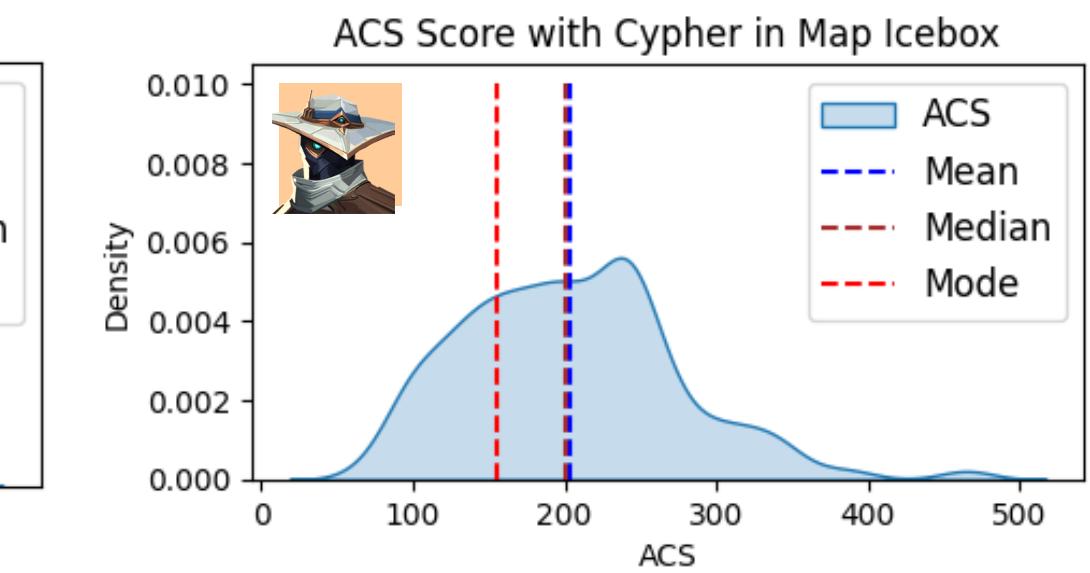
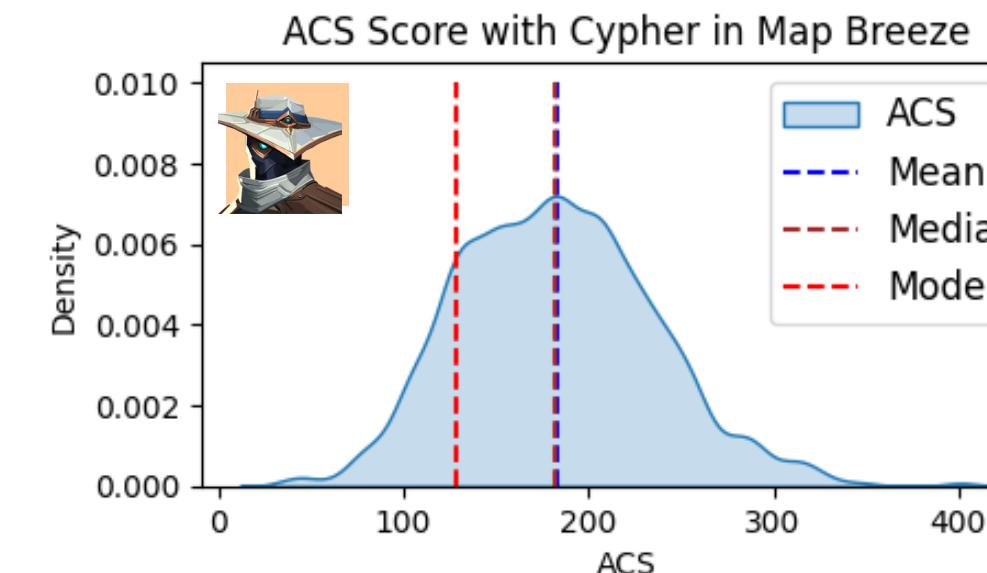
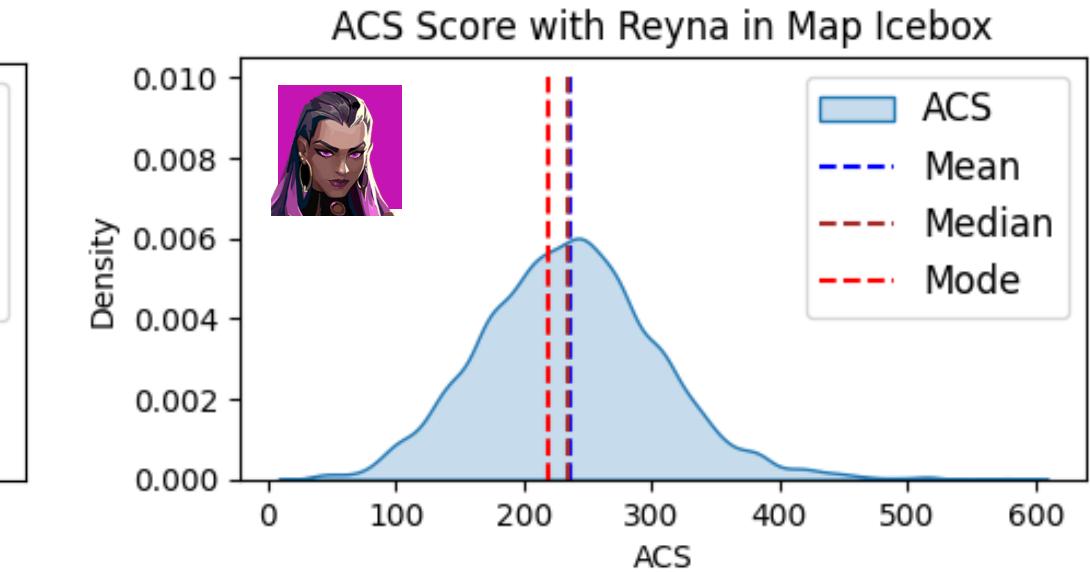
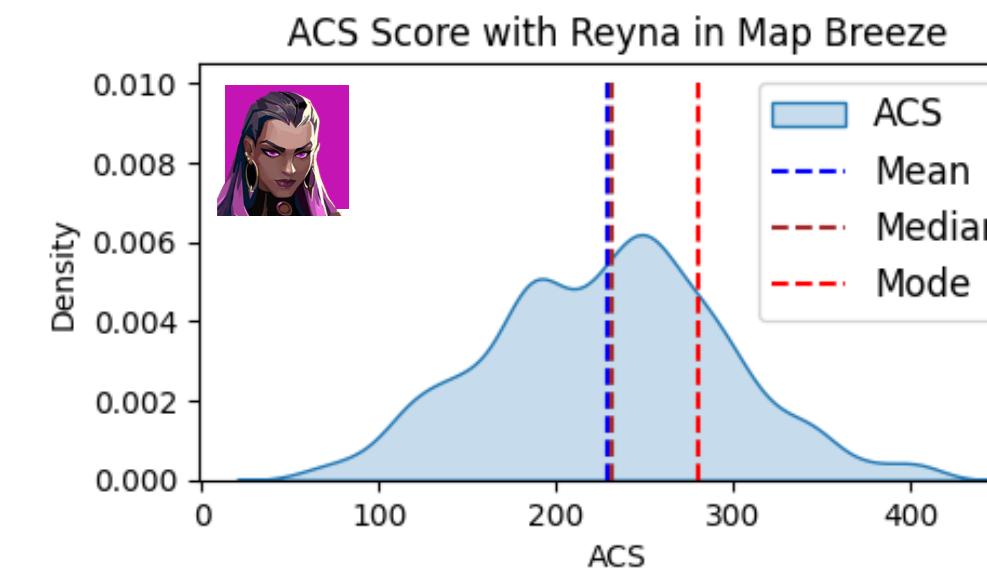


# Persebaran Nilai ACS pada setiap Agent (4)





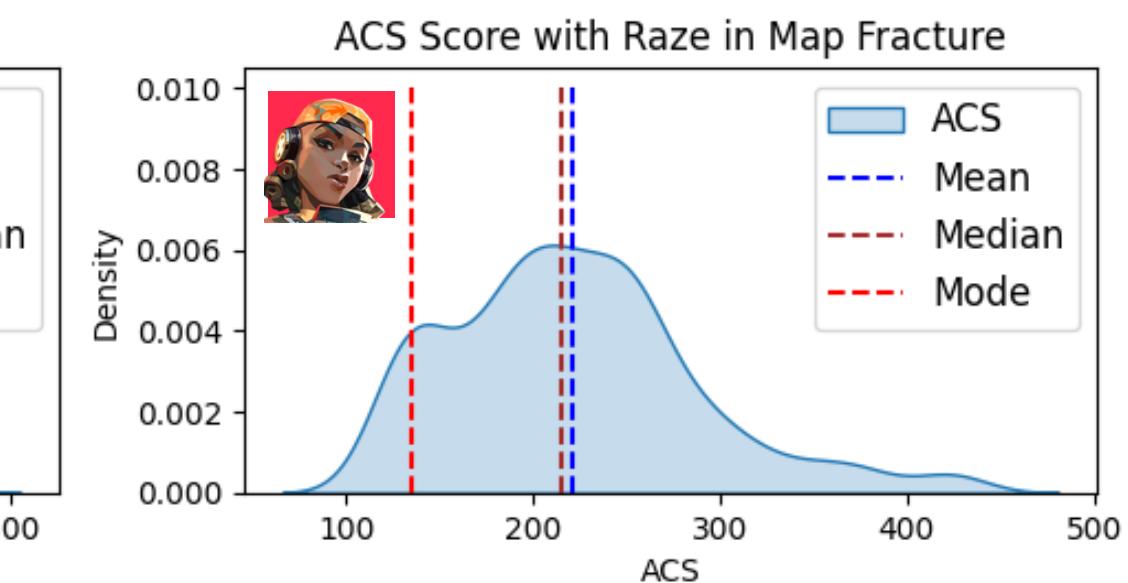
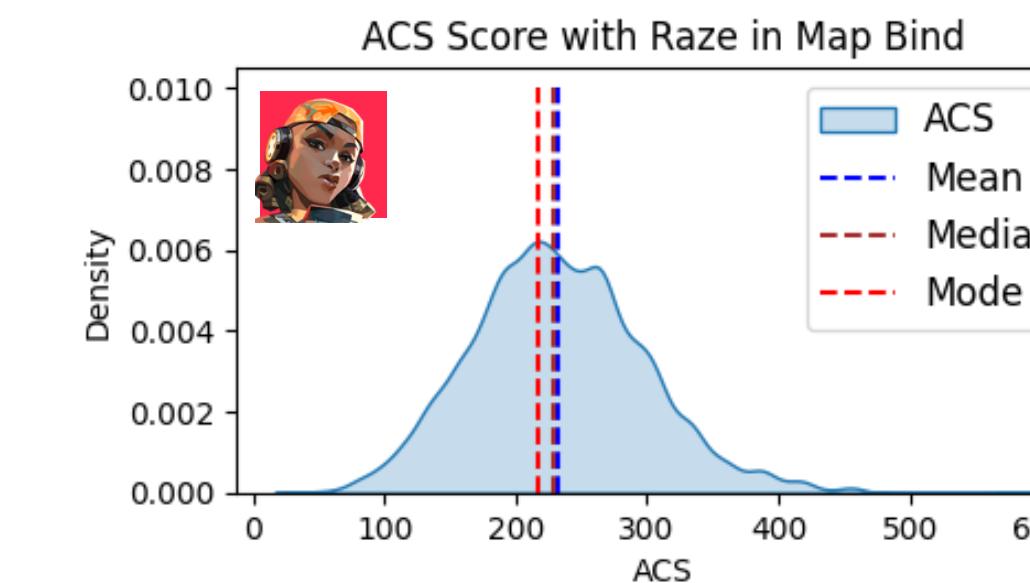
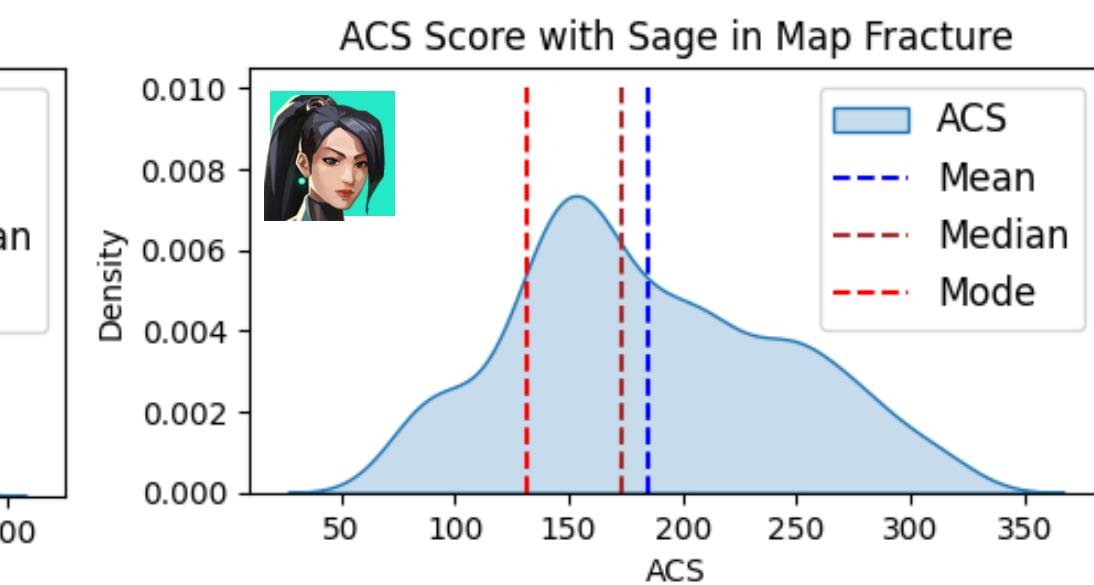
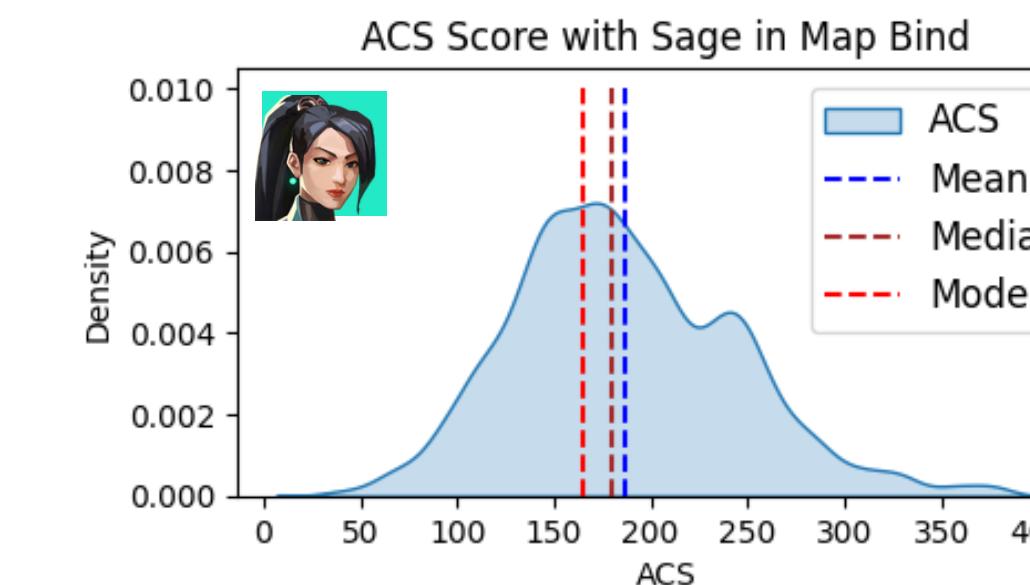
# Contoh Perubahan Persebaran Nilai ACS untuk setiap Agent Berdasarkan Map (I)



Terlihat ada beberapa Agent yang mengalami perubahan nilai ACS pada map Breeze dan Icebox

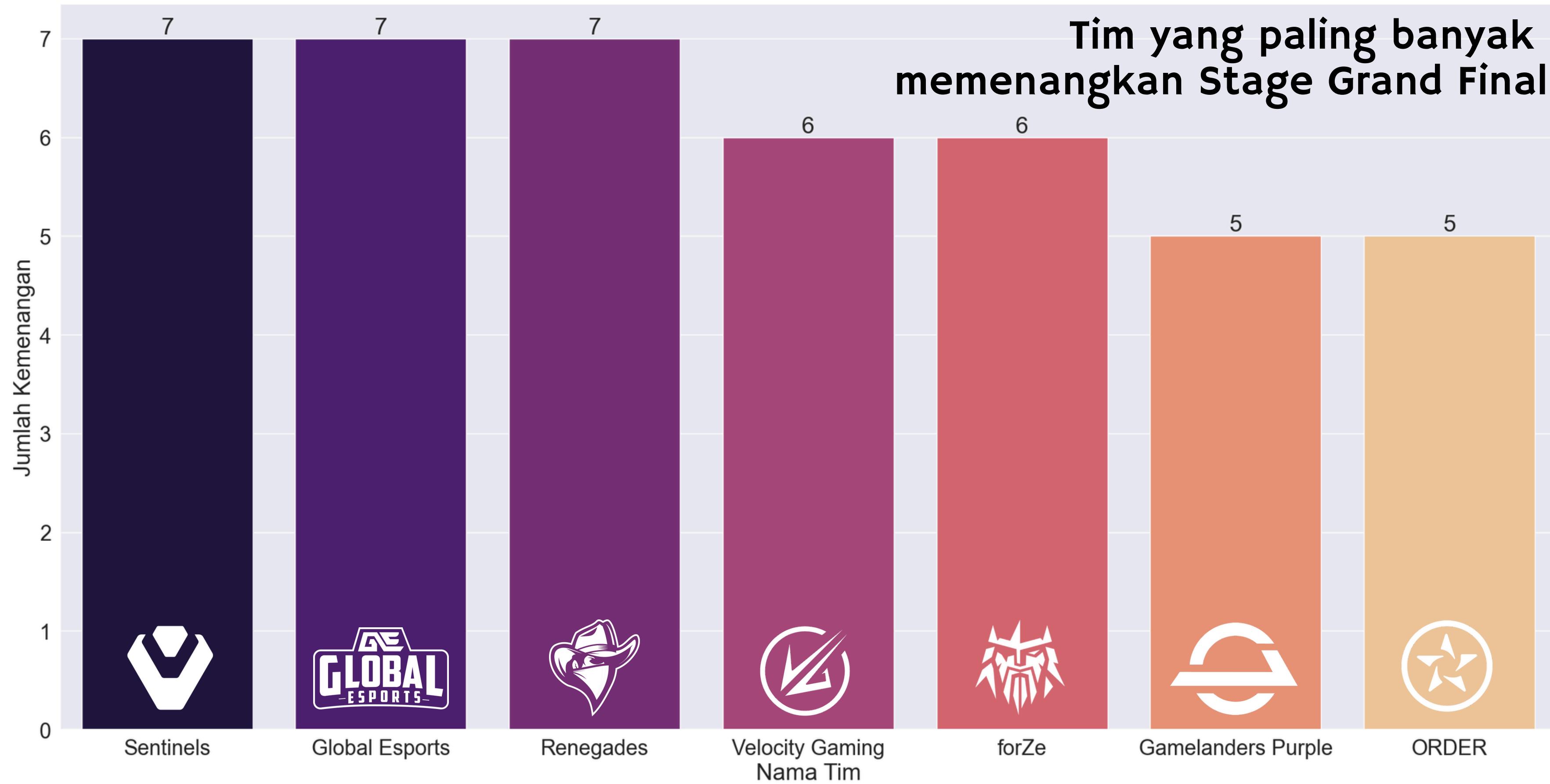


## Contoh Perubahan Persebaran Nilai ACS untuk setiap Agent Berdasarkan Map (2)



Terlihat ada beberapa Agent yang mengalami perubahan nilai ACS pada map Bind dan Fracture

# Tim yang paling banyak memenangkan Stage Grand Final



# Top 3 Komposisi Agent yang Paling Banyak Memenangkan Pertandingan



Cypher      Omen      Jett      Raze      Sova

417 Kemenangan



Cypher      Omen      Jett      Phoenix      Sova

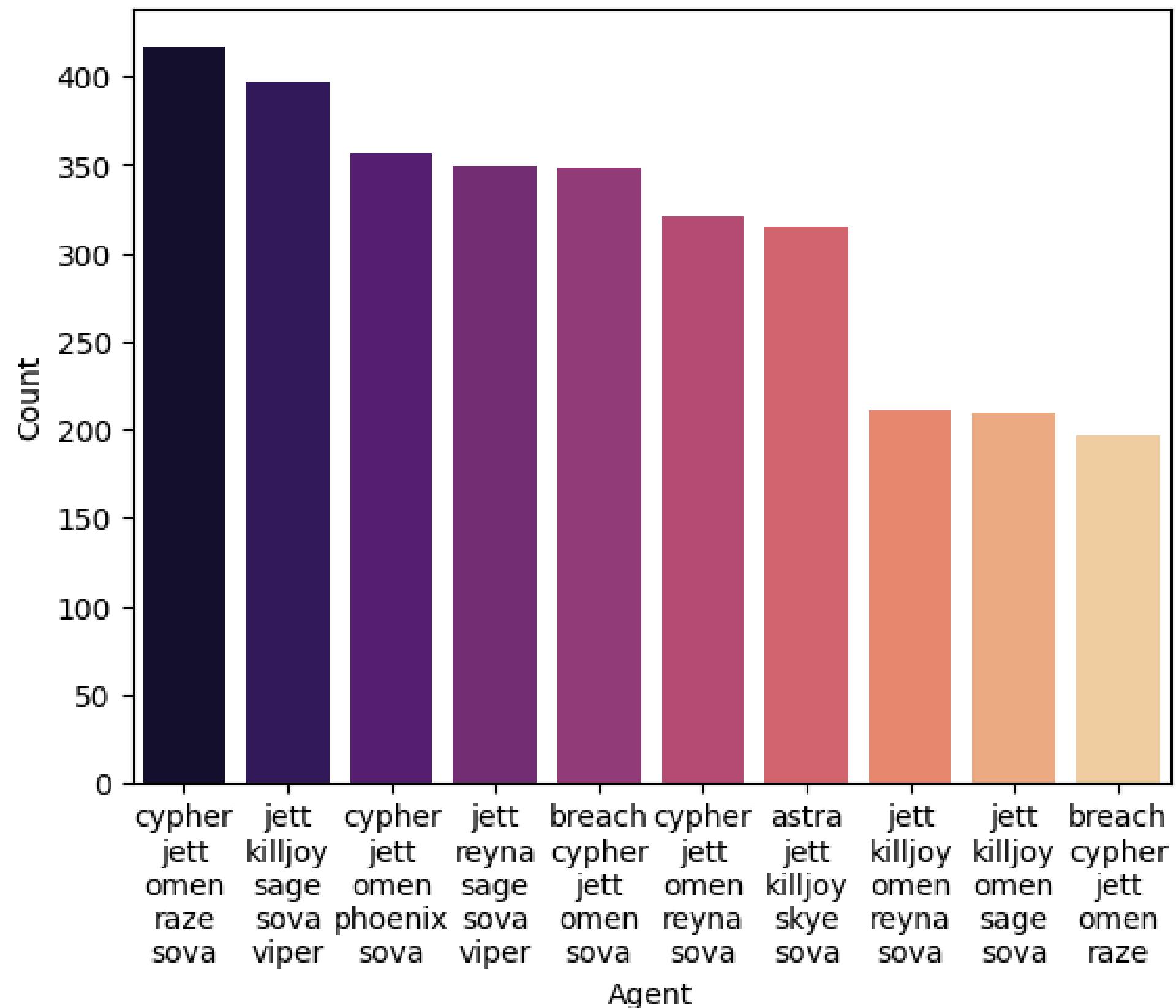
356 Kemenangan



Jett      Killjoy      Sage      Sova      Viper

397 Kemenangan

# Top 10 Komposisi Agent yang Paling Banyak Memenangkan Pertandingan



- Setiap komposisi tim memiliki setidaknya satu agent dari setiap role (duelist, controller, initiator, dan sentinel).
- Jett merupakan duelist yang ada pada semua komposisi top 10 karena dibandingkan dengan duelist lainnya, ia sangatlah lincah.
- Sova merupakan initiator yang ada pada 9 dari 10 komposisi top 10 karena merupakan agent yang serba bisa. Ia dapat berperan sebagai pelacak musuh, penyerang, ataupun penjaga.
- Alasan lainnya, pada waktu dimana dataset didapatkan, belum ada initiator yang bisa melacak musuh selain Sova.

## Eksplorasi Mandiri I: Rata-Rata Kill/Death setiap Agent per Map

### Tujuan:

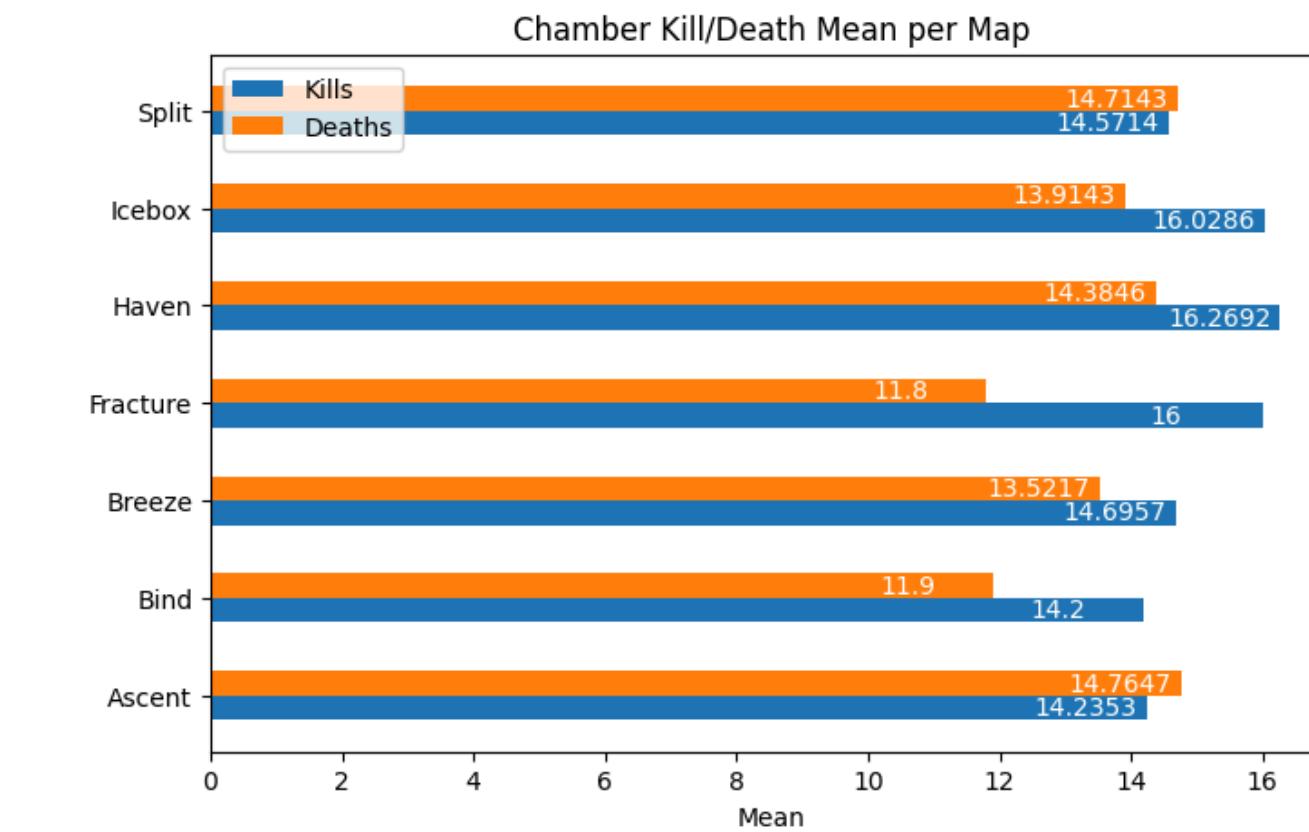
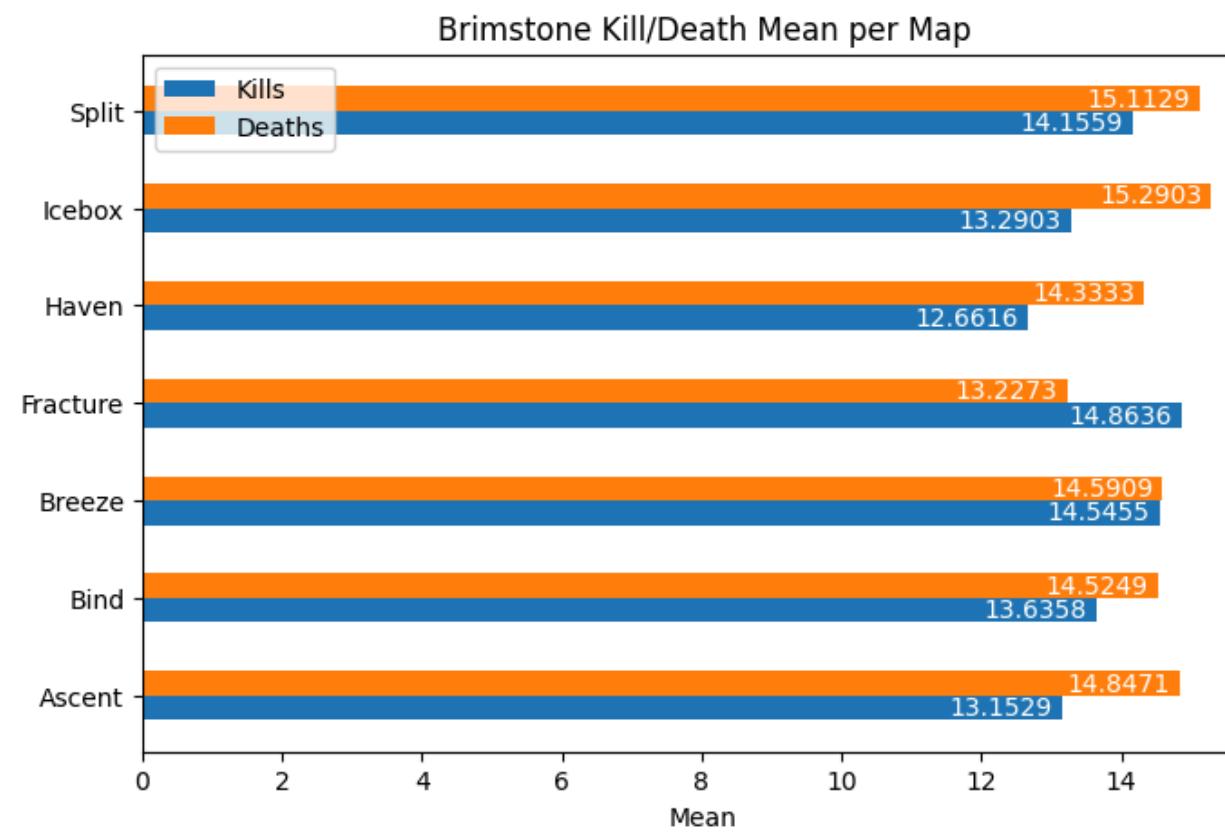
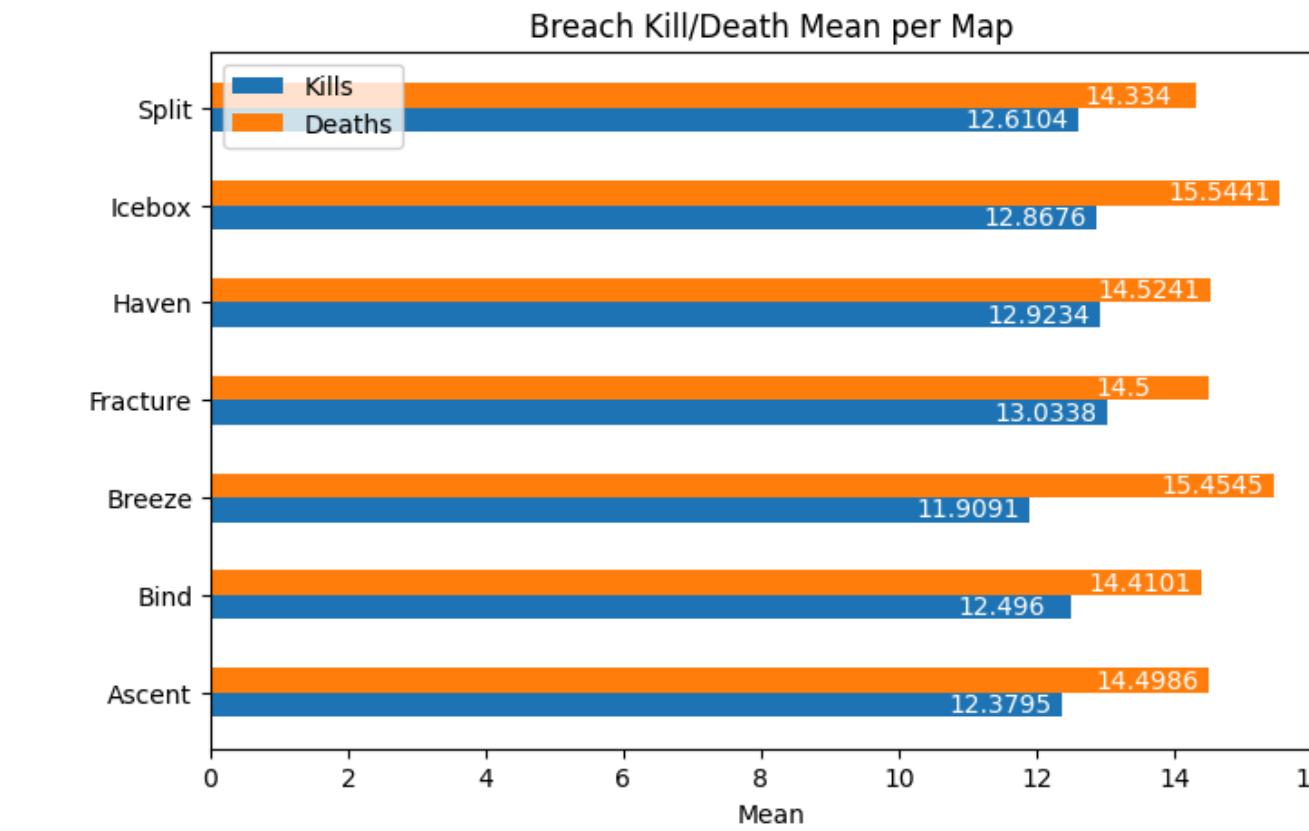
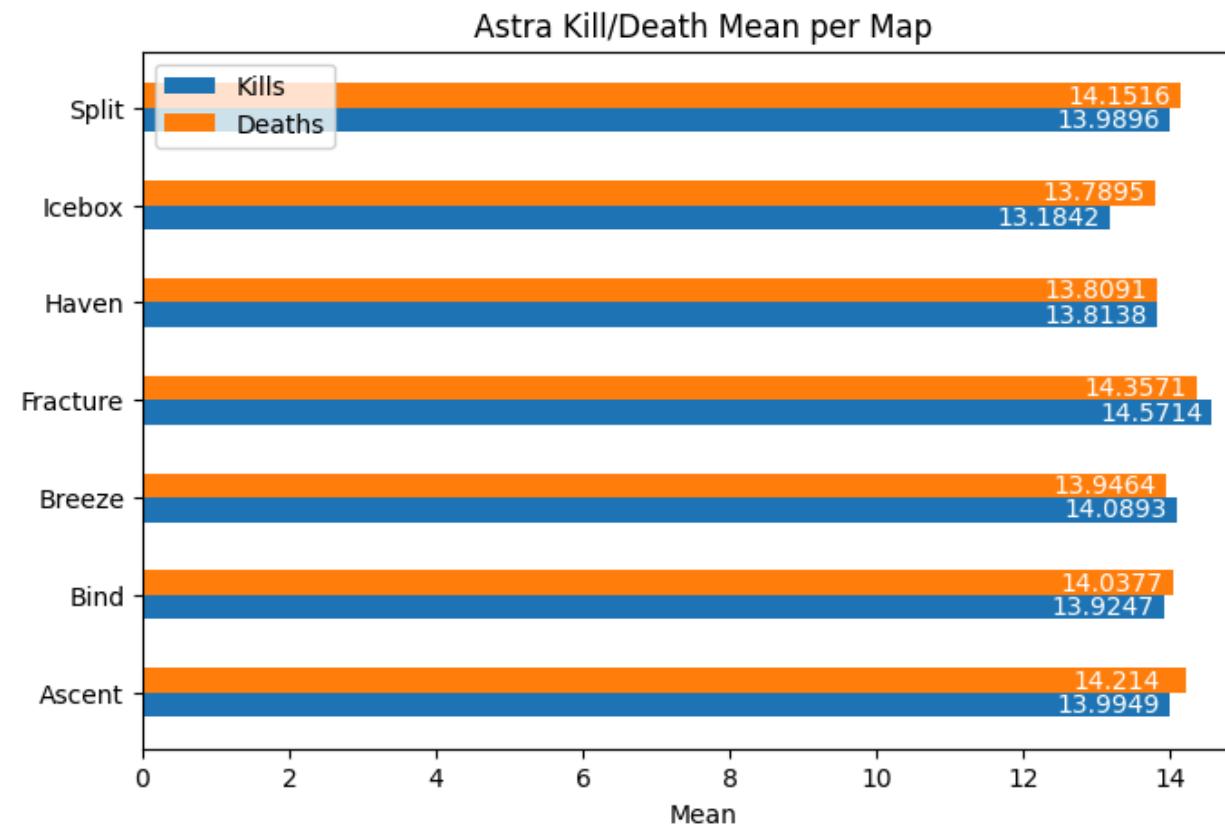
Mengetahui informasi Kill/Death dari setiap Agent untuk semua Map.

### Kegunaan:

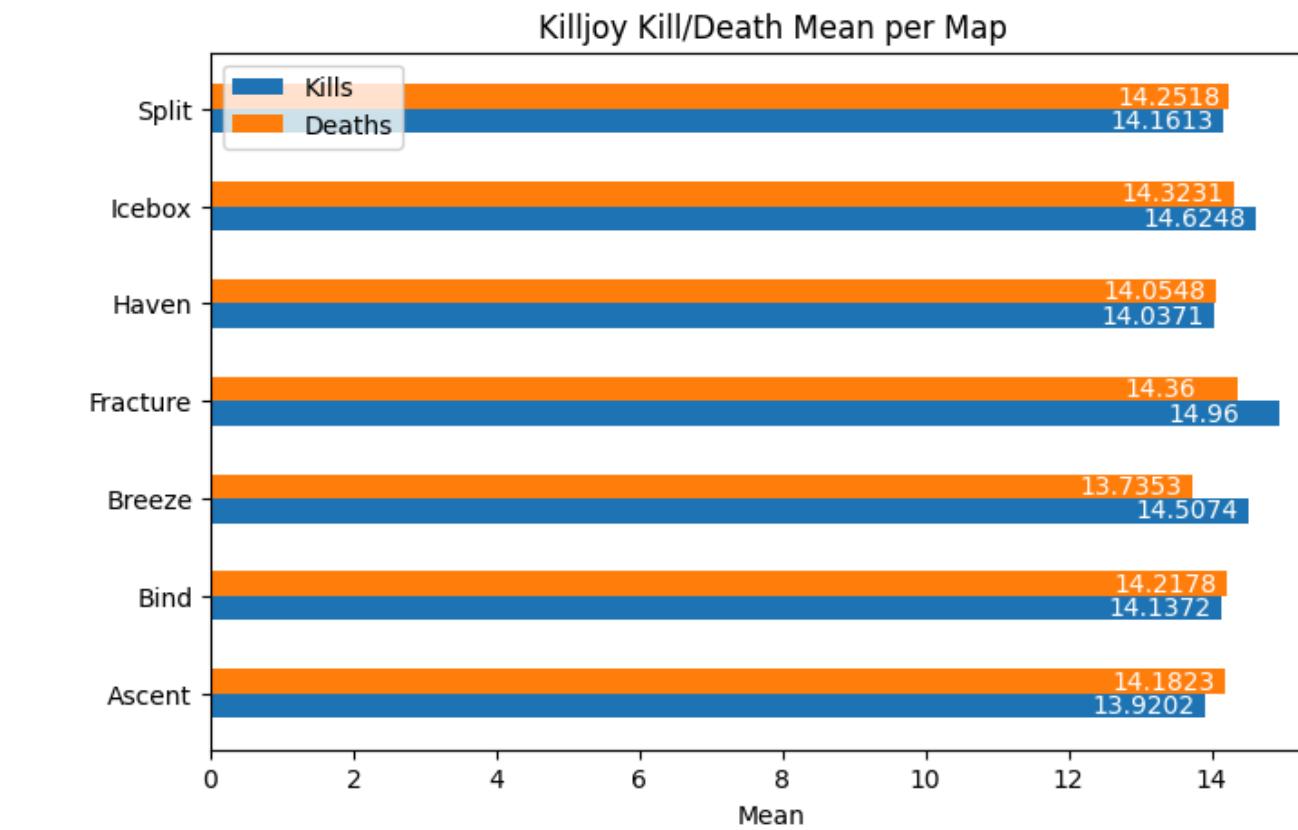
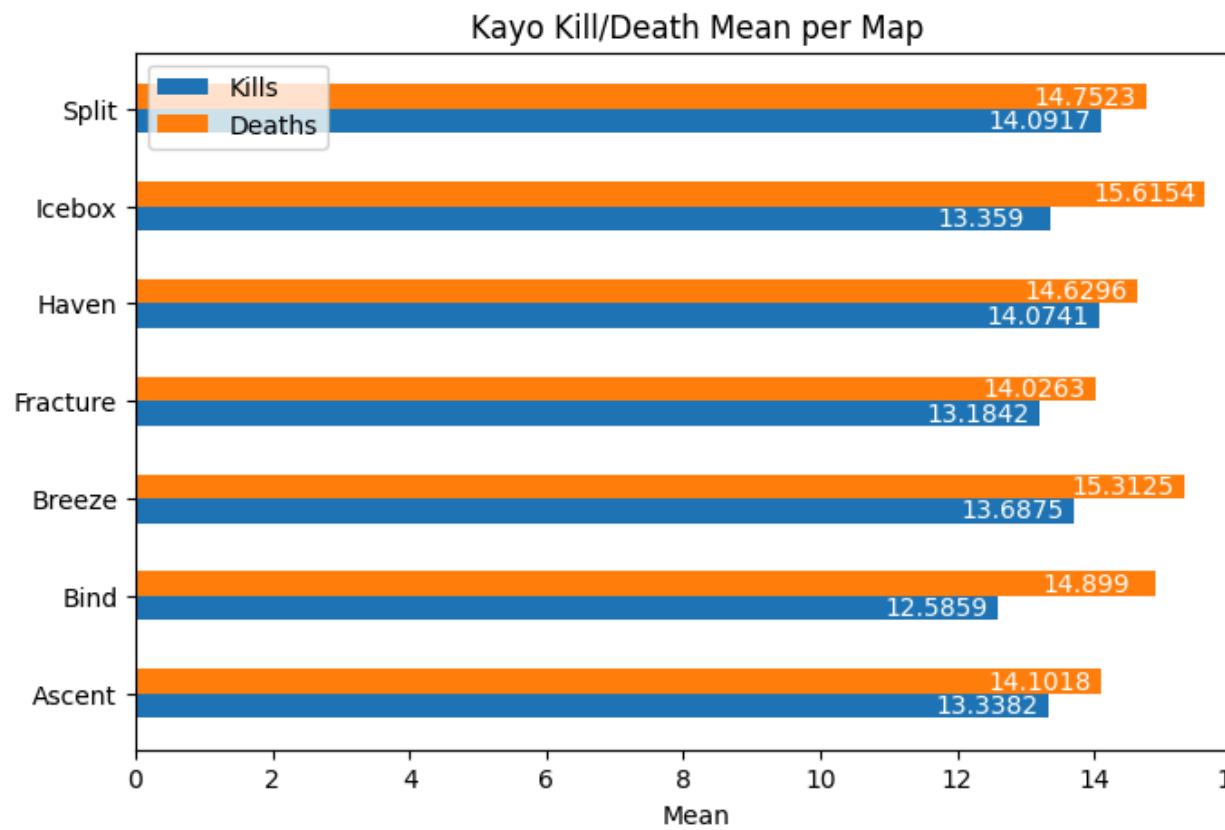
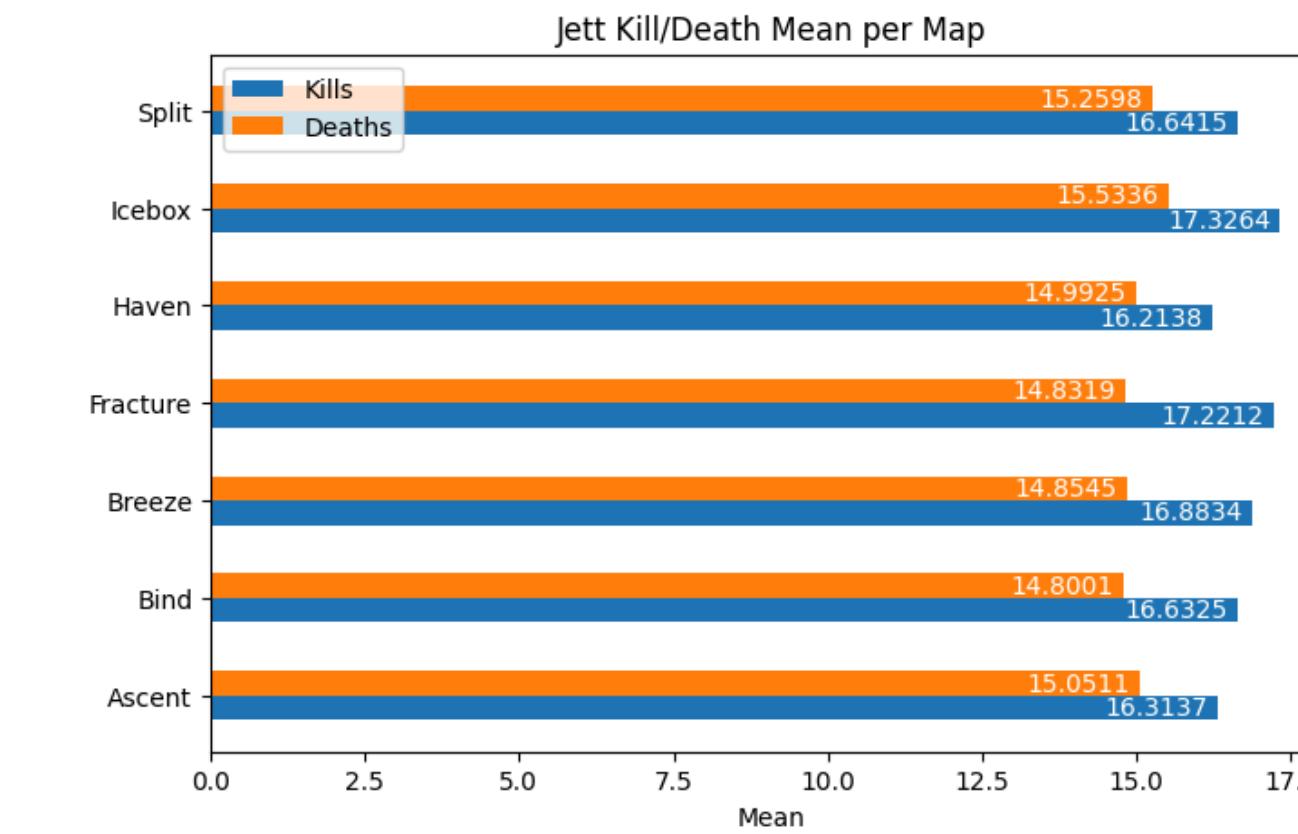
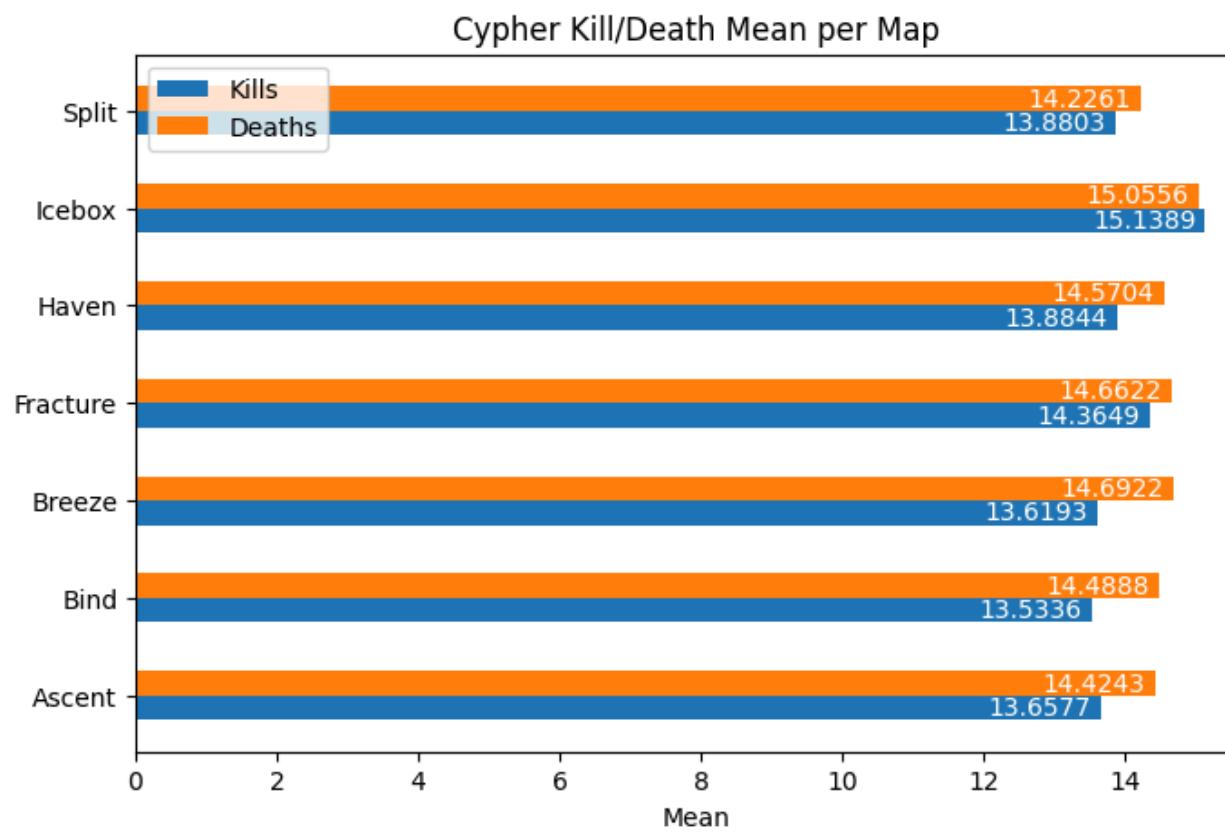
Dapat membantu Developer memilih Agent yang akan ditingkatkan berdasarkan K/D pada Map tertentu.



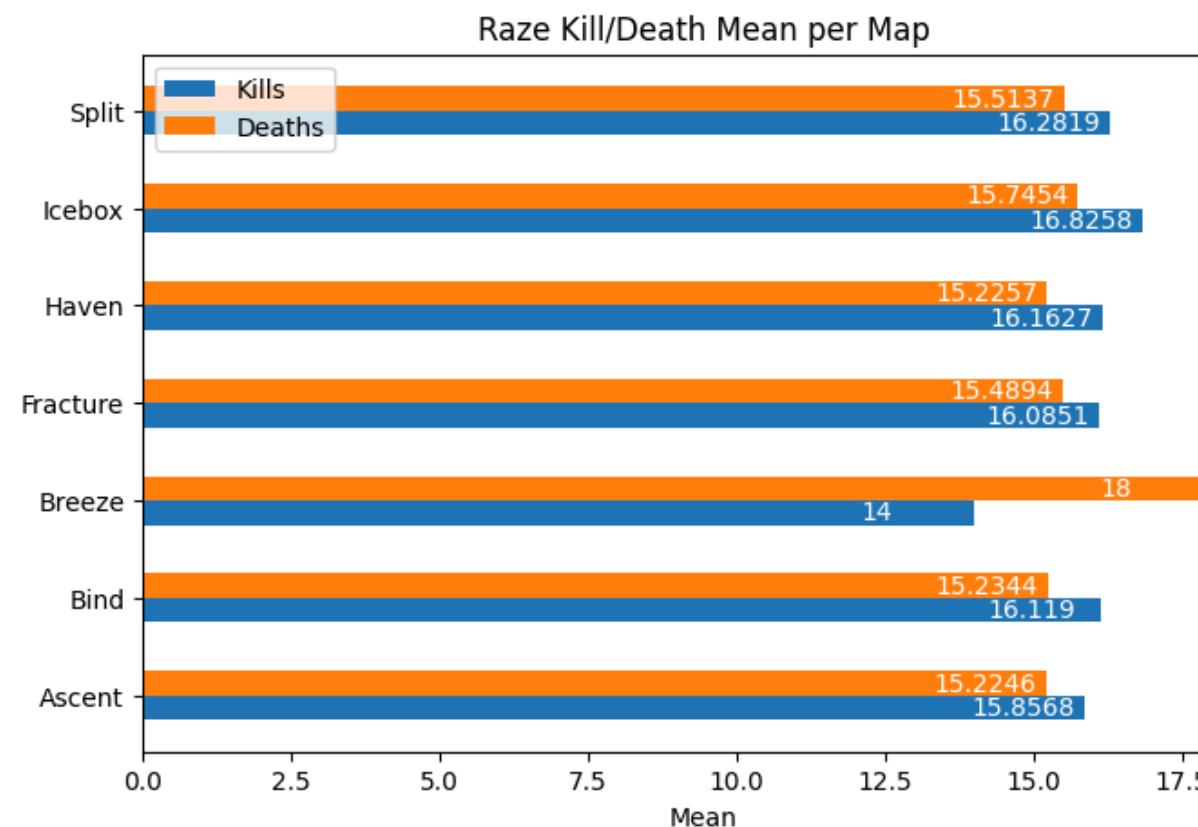
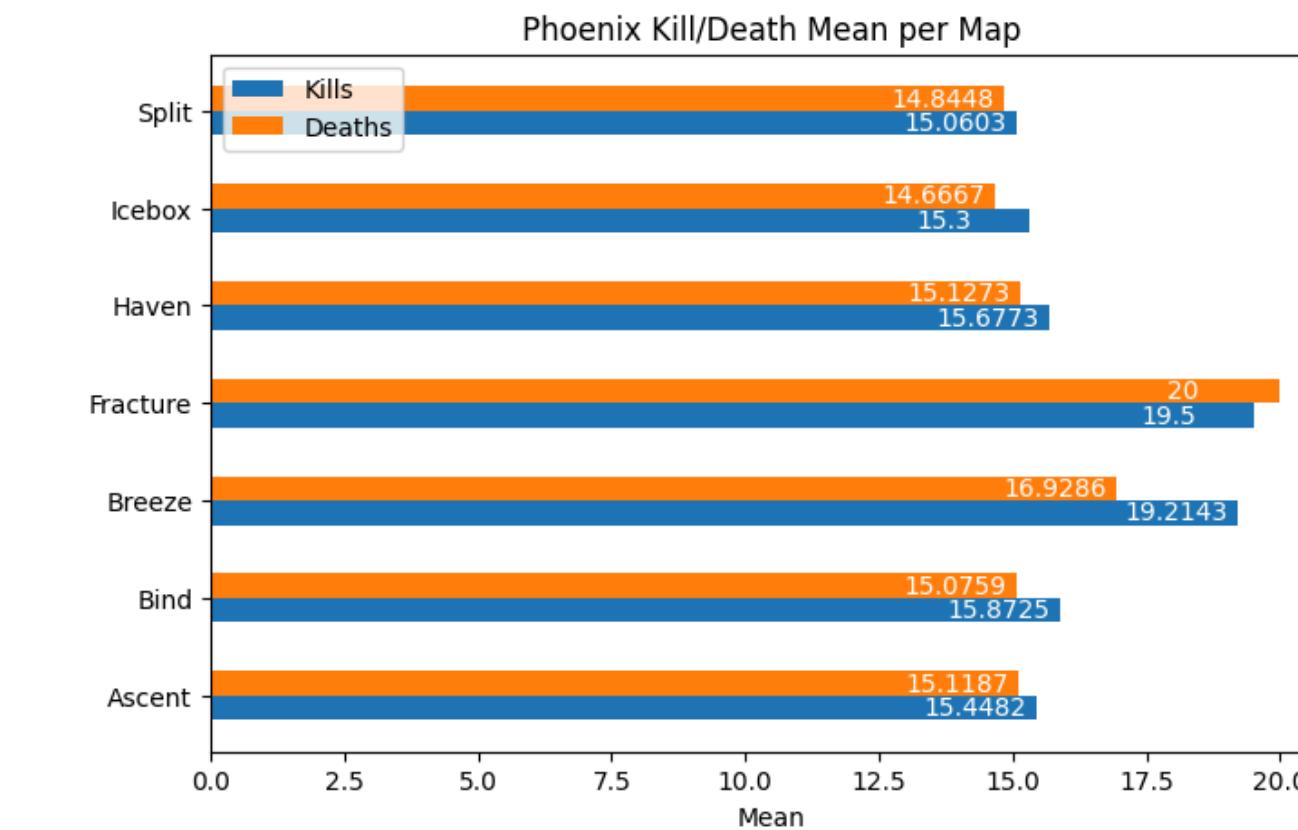
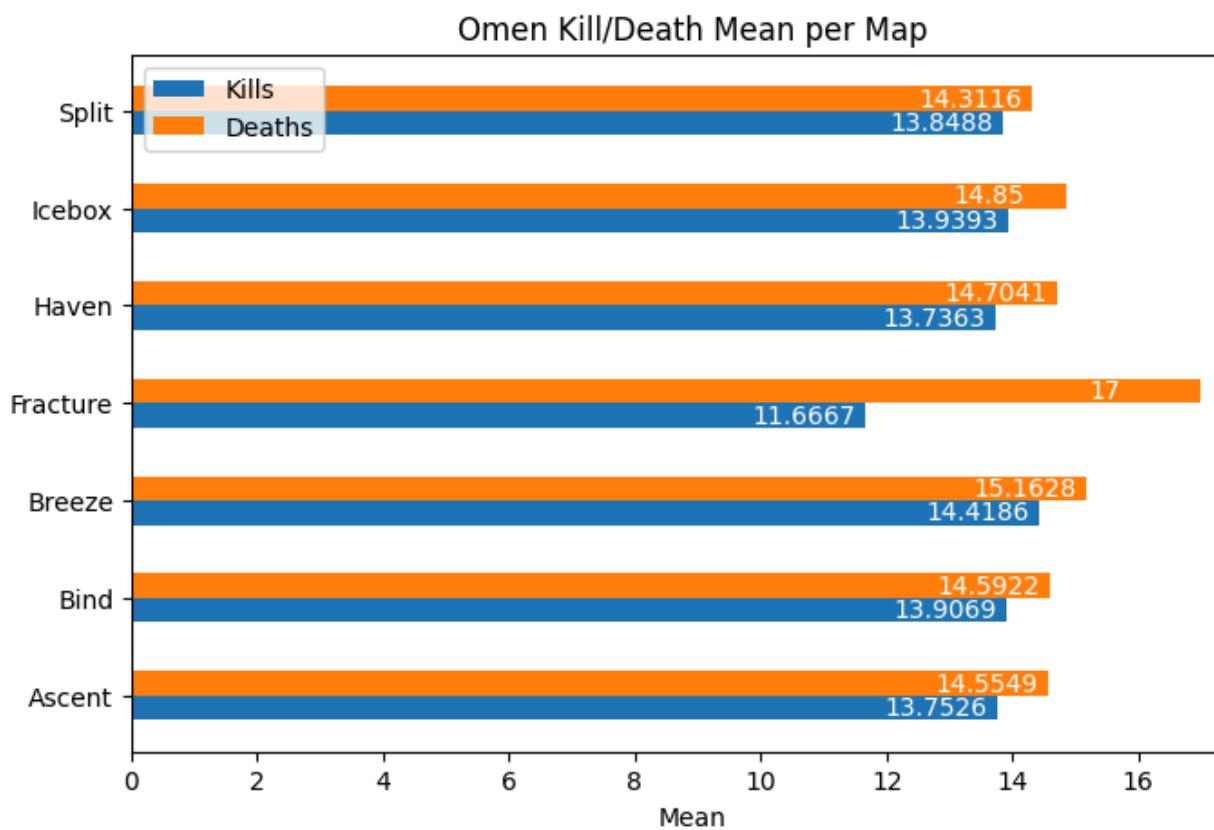
# Eksplorasi Mandiri I: Rata-Rata Kill/Death setiap Agent per Map (I)



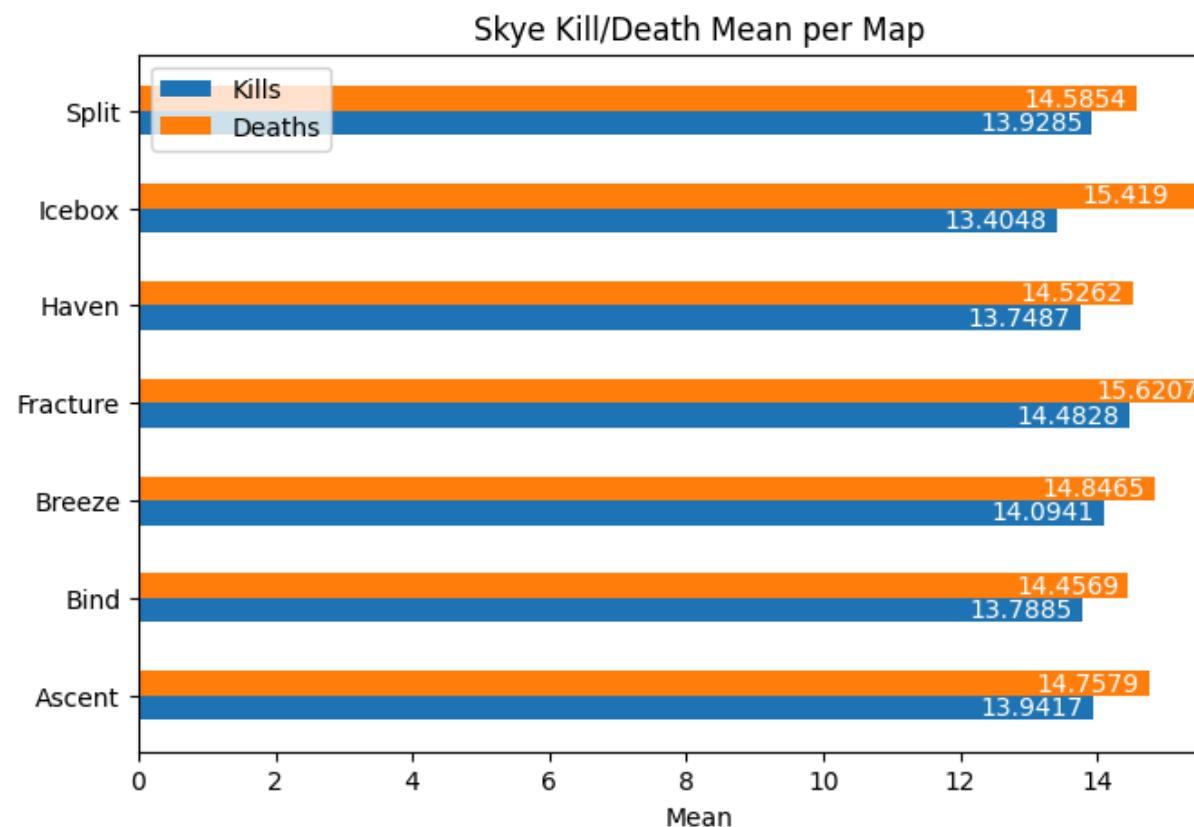
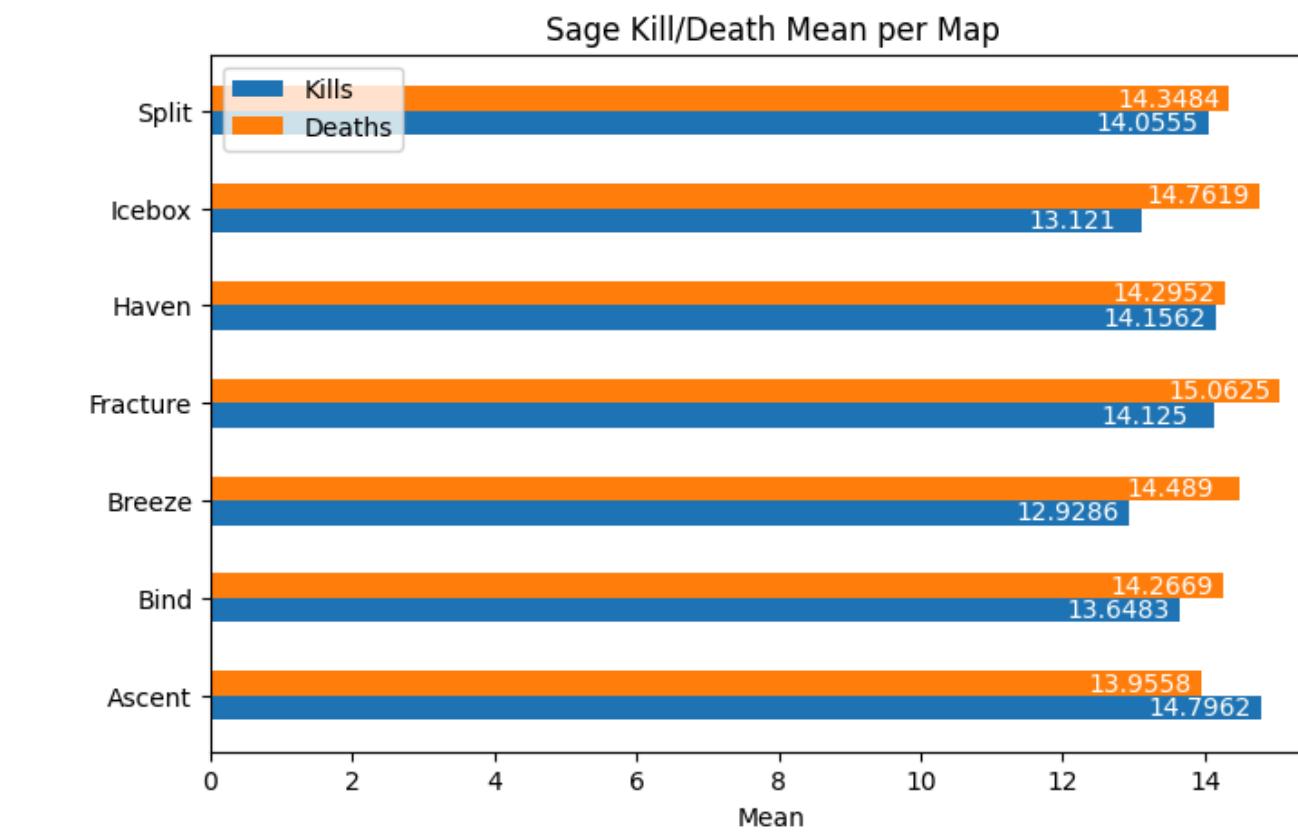
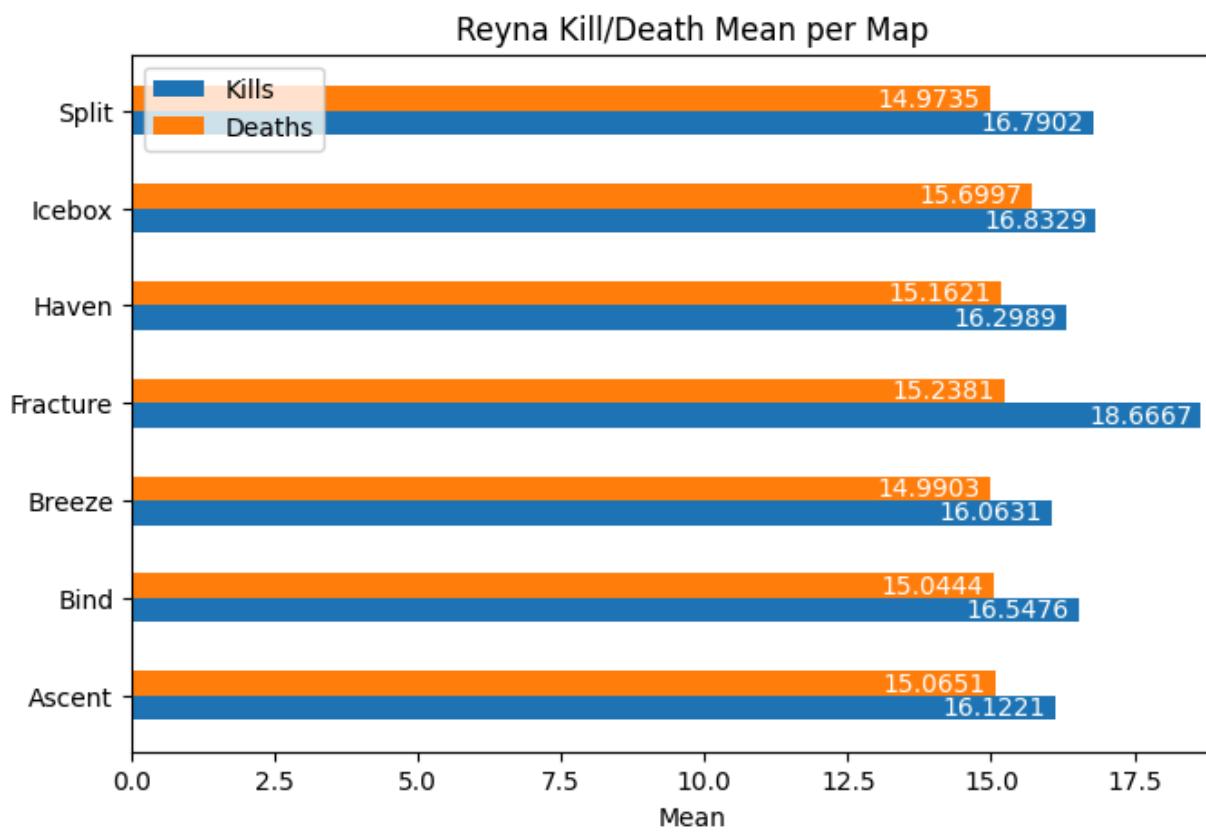
# Eksplorasi Mandiri I: Rata-Rata Kill/Death setiap Agent per Map (2)



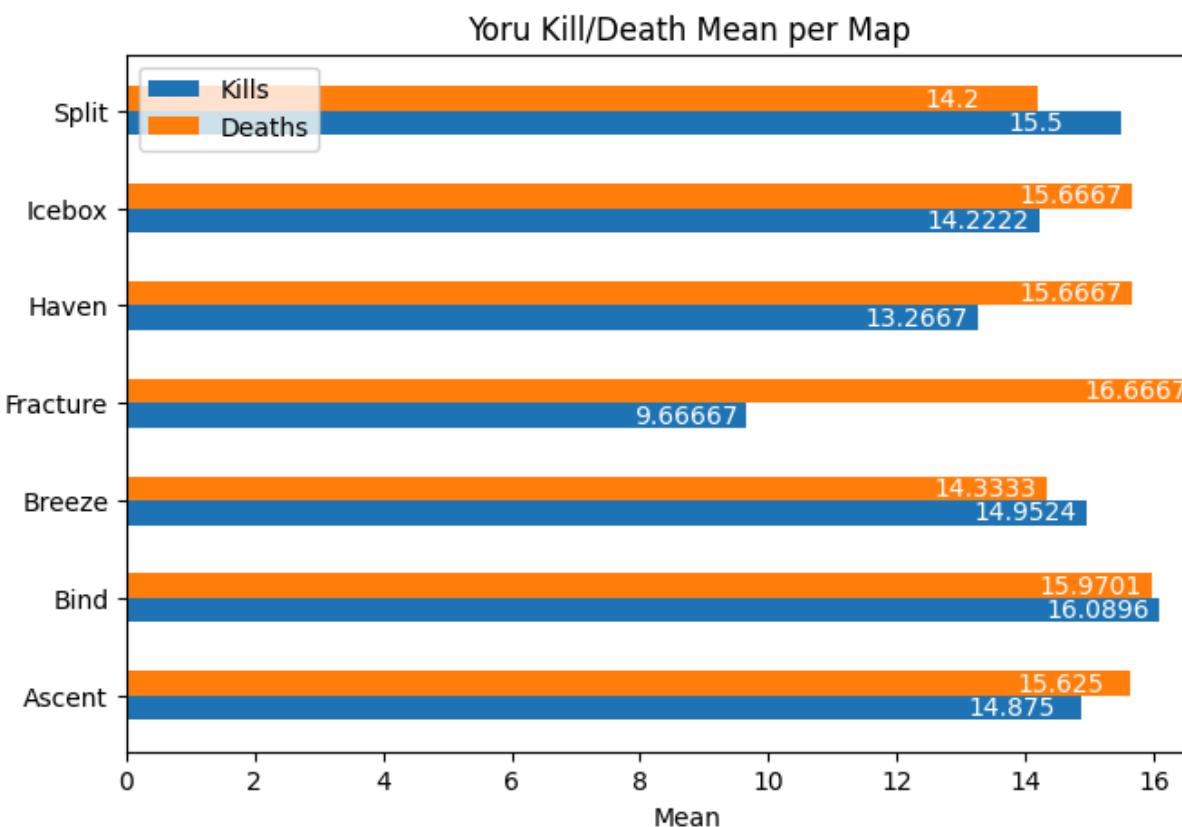
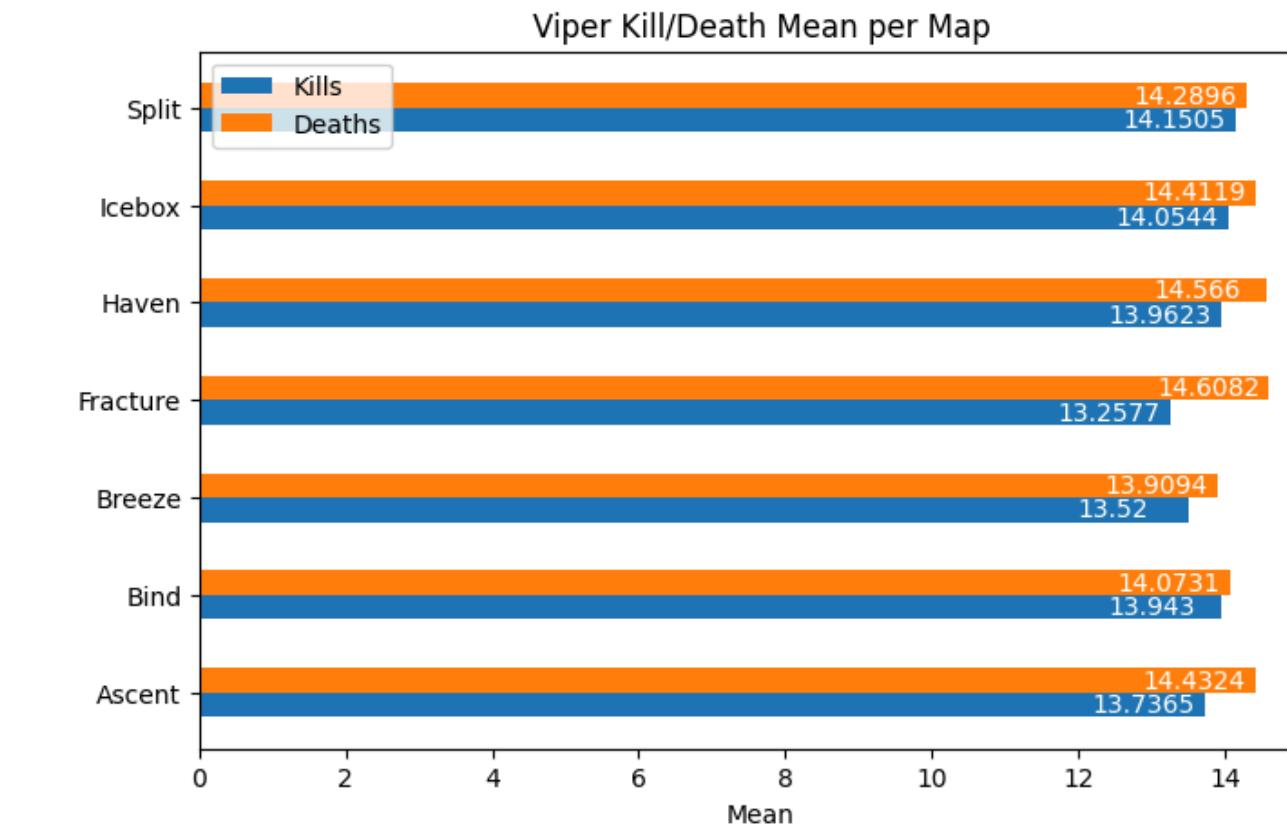
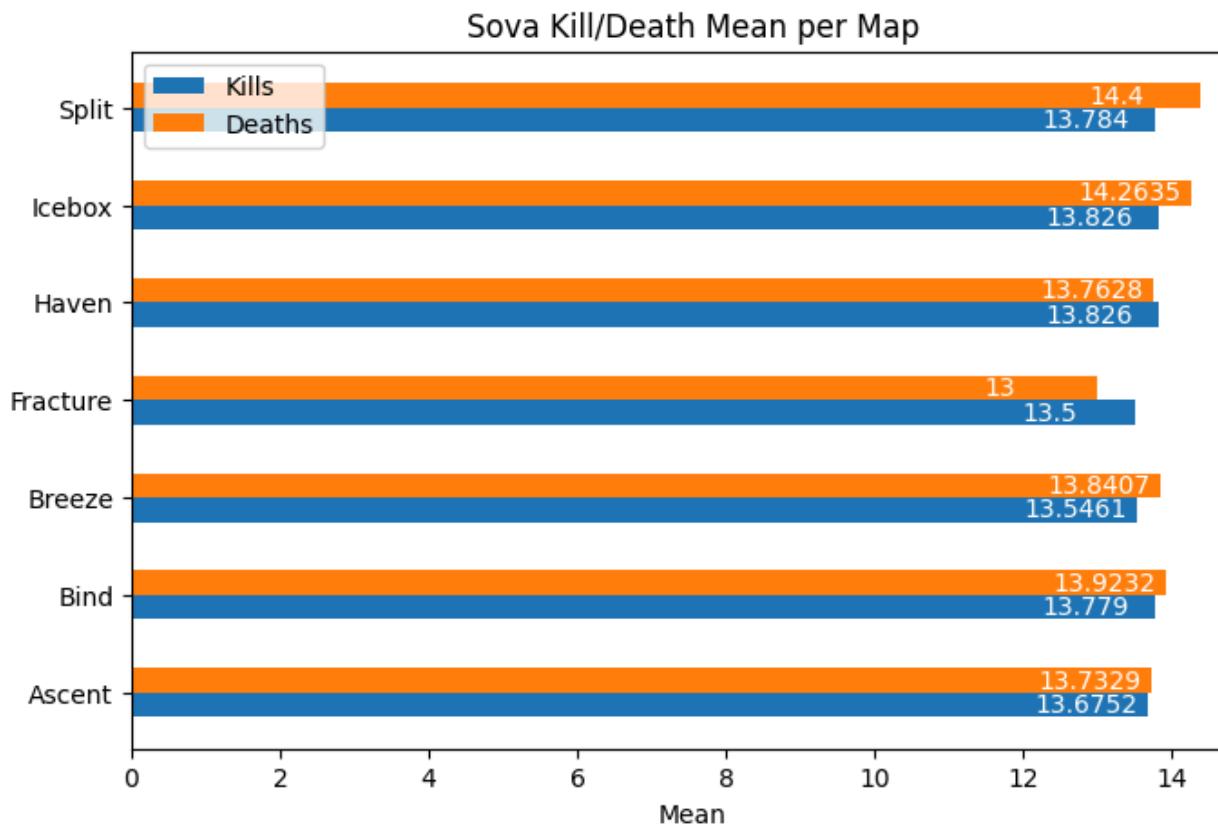
# Eksplorasi Mandiri I: Rata-Rata Kill/Death setiap Agent per Map (3)



# Eksplorasi Mandiri I: Rata-Rata Kill/Death setiap Agent per Map (4)



# Eksplorasi Mandiri I: Rata-Rata Kill/Death setiap Agent per Map (5)



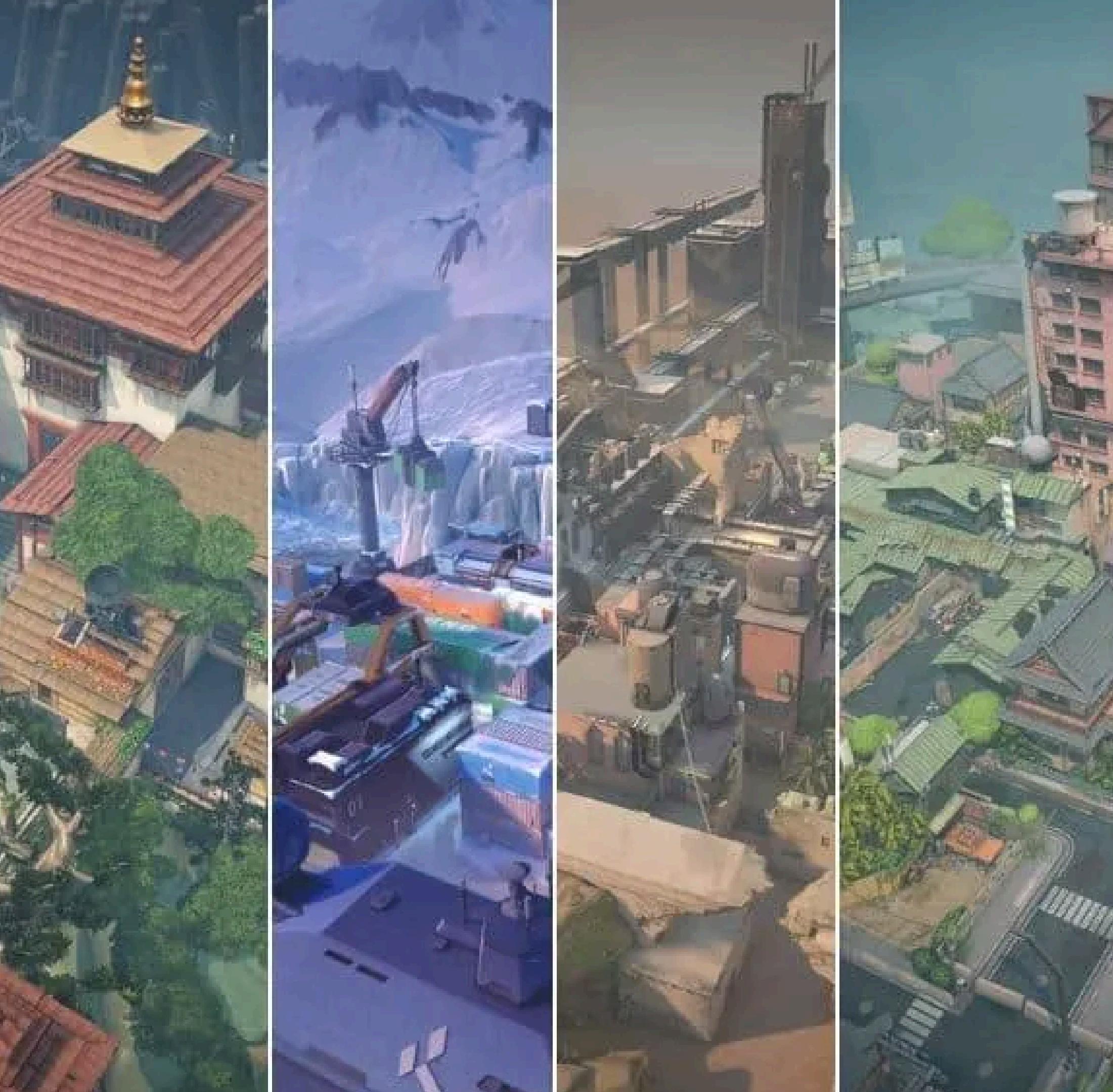
# **Eksplorasi Mandiri II: Map yang paling sering digunakan di setiap Patch**

## **Tujuan:**

Mengetahui Map apa yang paling sering digunakan dalam turnamen Valorant di setiap Patch.

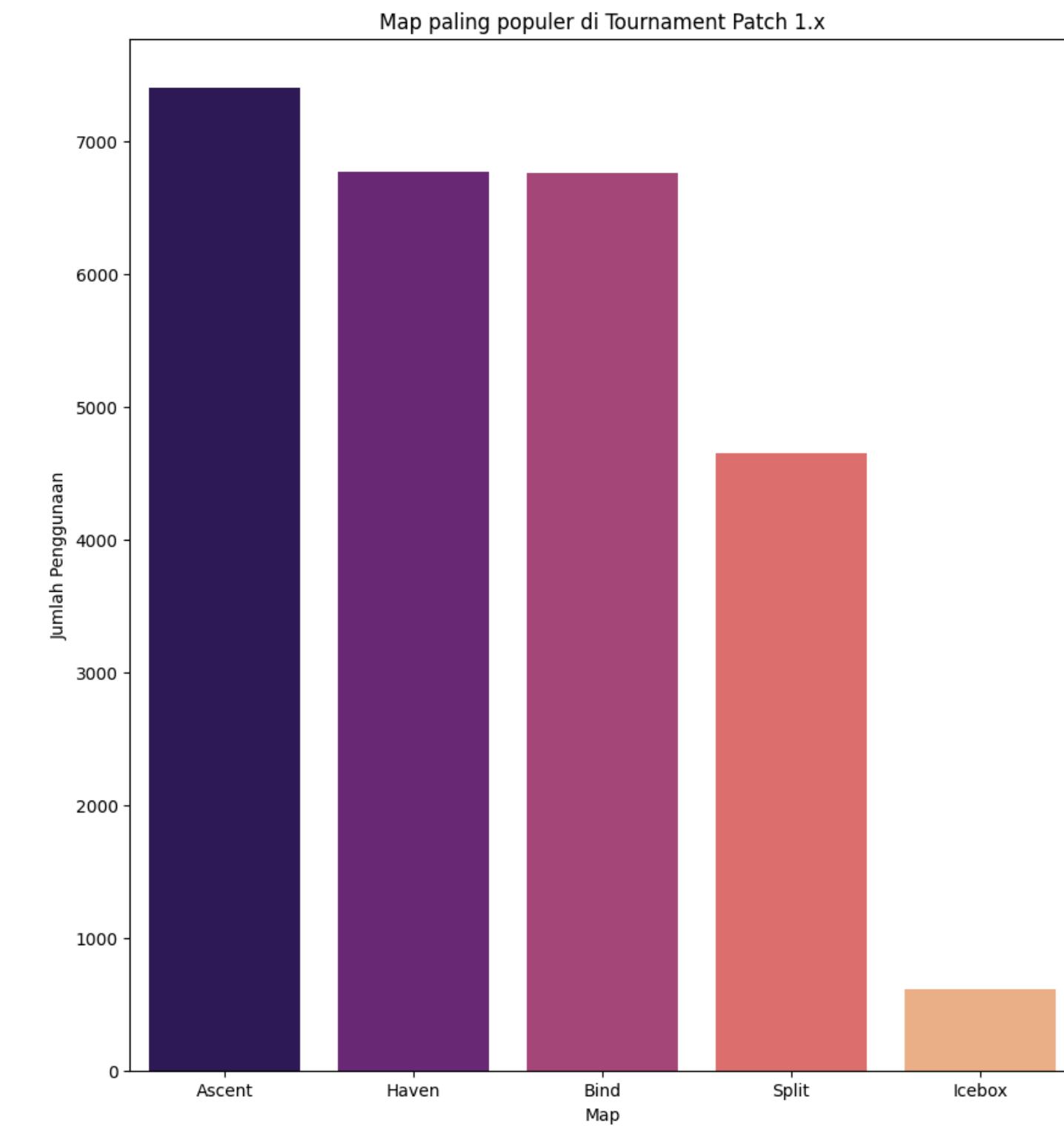
## **Kegunaan:**

Dapat digunakan oleh tim yang ingin berkompetisi untuk mempersiapkan diri dengan berlatih pada Map tersebut.



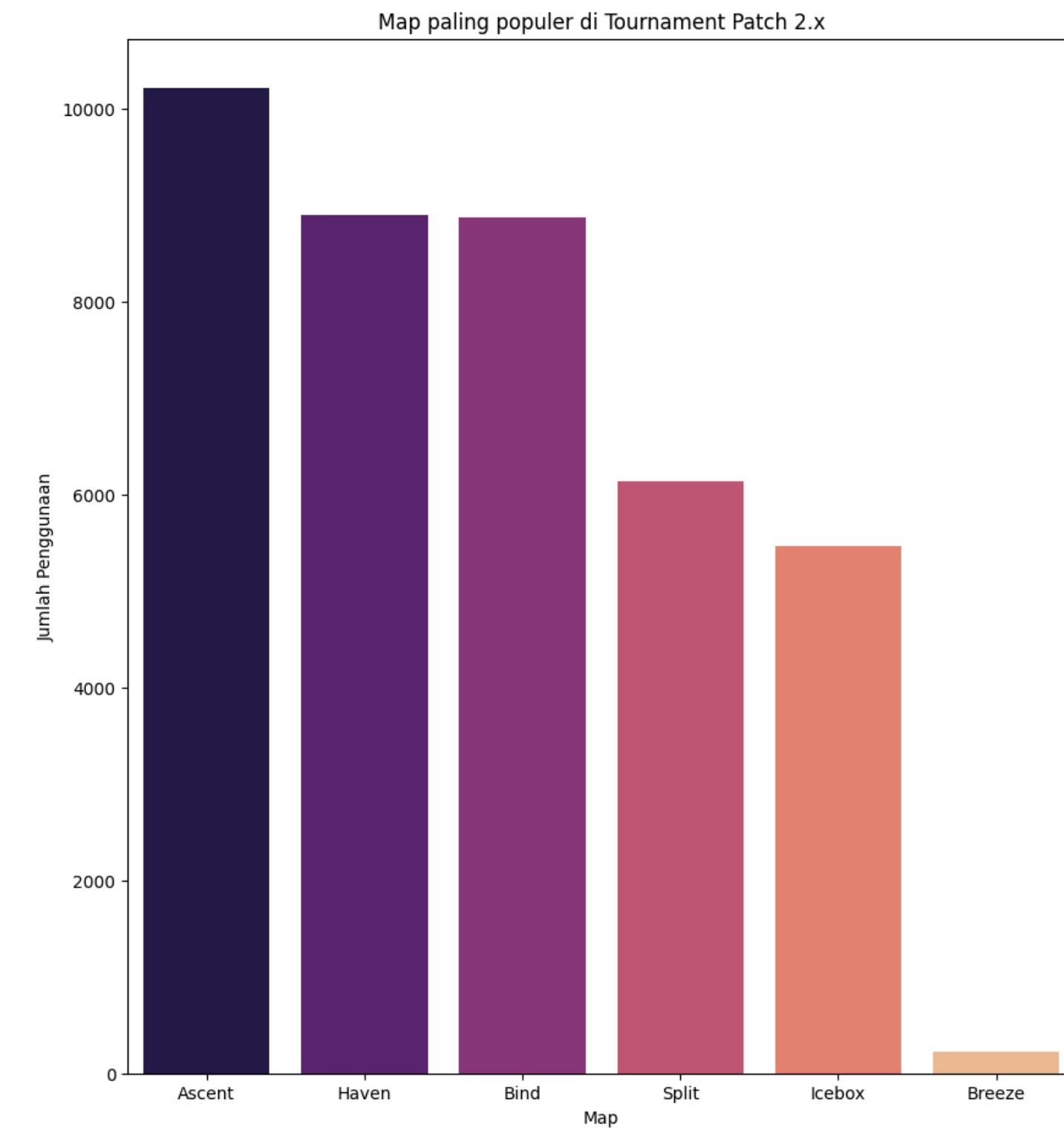


## Map yang paling sering digunakan di Patch 1.x



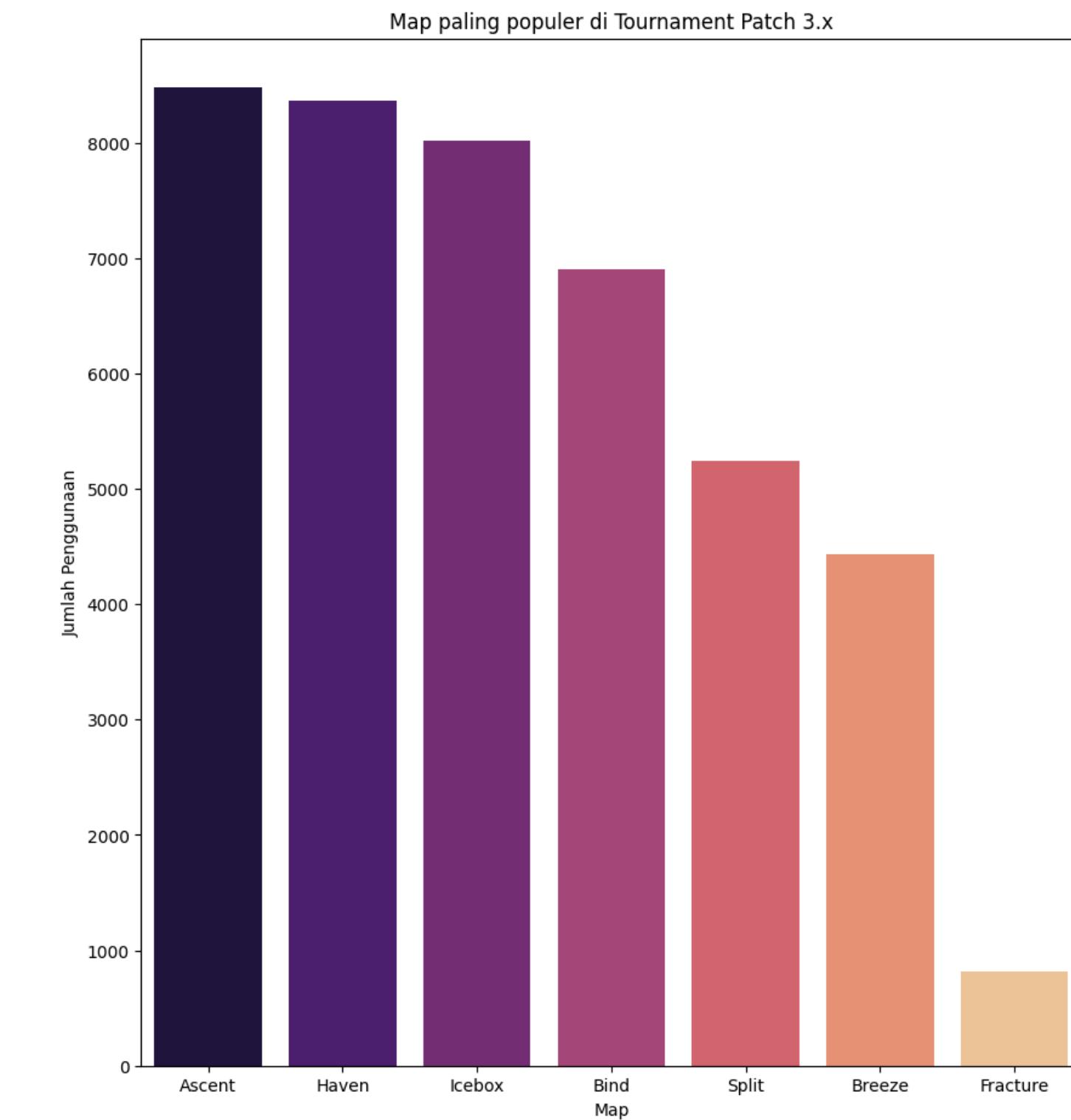


## Map yang paling sering digunakan di Patch 2.x





## Map yang paling sering digunakan di Patch 3.x



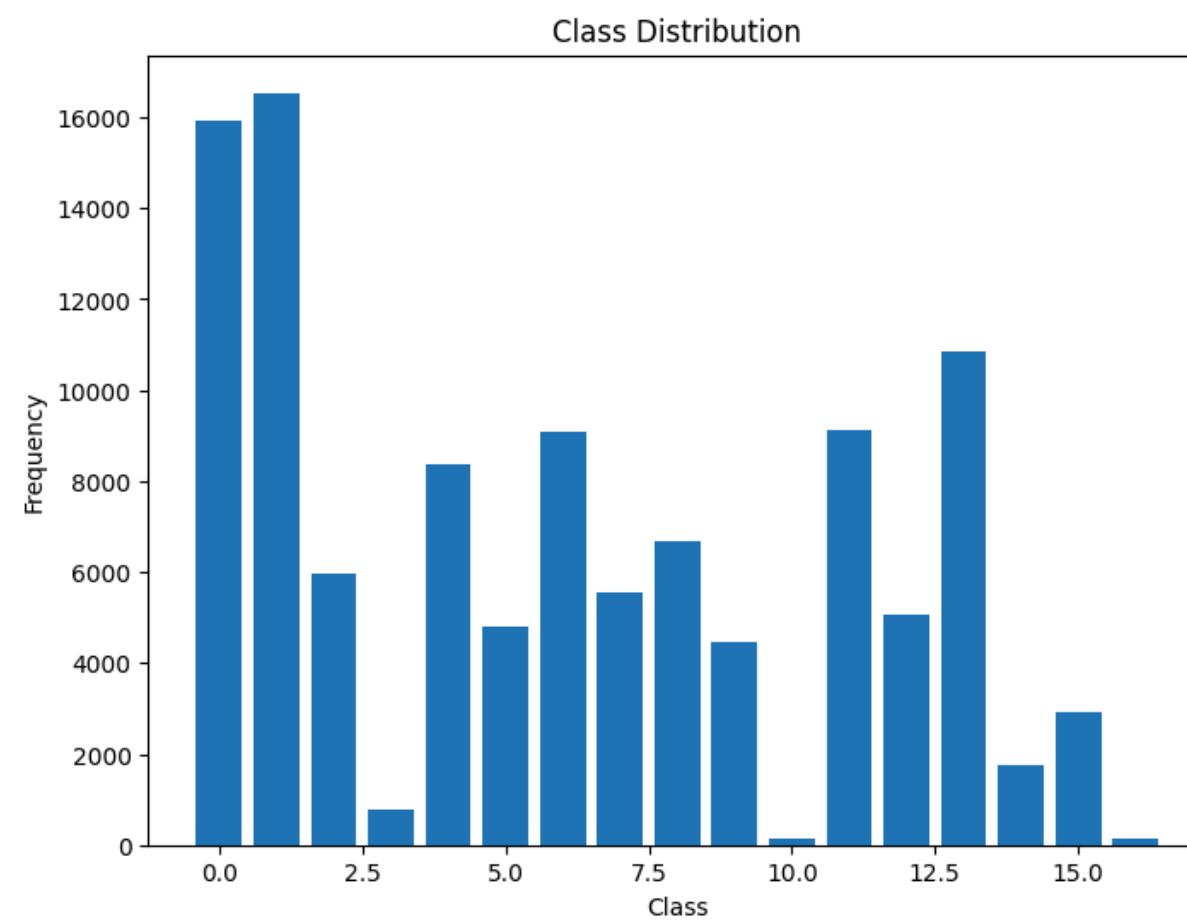
# Classification



# Catatan Awal

**Tujuan Klasifikasi :** Memprediksi jenis agent yang digunakan oleh setiap pemain

**Metrik Evaluasi :** F1-Micro Score, berfungsi menggabungkan presisi dan recall dari setiap kelas menjadi satu nilai tunggal, memberikan perhatian yang setara pada setiap kelas dan berguna saat menghadapi ketidakseimbangan kelas.



Class	Frequencies:
1	16539
0	15936
13	10845
11	9118
6	9089
4	8382
8	6684
2	5974
7	5564
12	5053
5	4812
9	4469
15	2938
14	1761
3	786
10	151
16	149

GameID	0.221821
EventID	0.218072
MatchID	0.215912
Patch	0.198414
Team2ID	0.145477
Team1ID	0.120366
Plants	0.093230
FirstKills	0.084458
Econ	0.082755
PlayerID	0.076499
FKFD_PlusMinus	0.069006
PlusMinus	0.058441
ADR	0.058302
Assists	0.053646
Kills	0.039397
Deaths	0.039237
Num_2Ks	0.026402
Defuses	0.023766
OnevOne	0.021817
Num_3Ks	0.021743
OnevTwo	0.019915
HS_Percent	0.018986
Num_4Ks	0.018076
Team1_MapScore	0.012734
Num_5Ks	0.009079
Team2_Eco	0.008194
Team2_SemiBuy	0.007816
OnevThree	0.005883
OnevFour	0.005112
Team1_Eco	0.004893
Team1_SemiBuy	0.003512
Team2_TotalRounds	0.003027
Team1_SemiEco	0.002629
Team2_SemiEco	0.002370
Team1_FullBuy	0.002328
OnevFive	0.002018
Team2_FullBuy	0.001596
Winner	0.000907
Team2_MapScore	0.000876
FirstDeaths	0.000111
Team1_TotalRounds	0.000110

Terlihat bahwa beberapa kelas memiliki frekuensi yang jauh lebih tinggi daripada kelas lainnya yang menunjukkan ketidakseimbangan yang signifikan antara kelas-kelas tersebut.

Korelasi antar fitur-fitur terhadap target yaitu 'Agent' memiliki korelasi yang rendah

# Tahapan

## I. Pre-Processing

- Dataset yang digunakan adalah dataset VPM yang sudah dilakukan preprocessing pada tahap EDA, Kemudian, dilakukan merging antara **df\_final** dengan dataset **agent\_encode**.
- One-Hot Encoding pada fitur "Map".
- Kolom-kolom yang memiliki tipe Object, yaitu 'PlayerName', 'TeamAbbreviation', 'Team1', 'Team2', 'Date', 'EventName', 'EventStage', dihapus dari **df\_final**.

## 2. Melakukan train test split

- Dataset dibagi menjadi 70% untuk training set dan 30% untuk validation set.

## 3. Normalisasi Data menggunakan StandardScaler

## 4. Menggunakan berbagai macam model klasifikasi dan mengevaluasi performance-nya

## 5. Melakukan resample data pada data training dengan berbagai macam metode oversampling dan undersampling, serta mengevaluasi performance-nya

## 7. Melakukan prediksi terhadap dataset test

## 6. Melatih model menggunakan data yang telah diresample dengan metode terbaik

# Model Selection (I)

Melatih 8 model klasifikasi berbeda dengan data preprocessing

	Algorithm	Accuracy	F1 Macro	F1 Micro	Precision Macro	Precision Micro	Recall Macro	Recall Micro
0	LogisticRegression	0.384326	0.256025	0.384326	0.309499	0.384326	0.252183	0.384326
1	KNeighborsClassifier	0.23067	0.135312	0.23067	0.154701	0.23067	0.13547	0.23067
2	DecisionTreeClassifier	0.232363	0.164627	0.232363	0.164997	0.232363	0.164351	0.232363
3	RandomForestClassifier	0.351347	0.201022	0.351347	0.246151	0.351347	0.205284	0.351347
4	GaussianNB	0.059985	0.05534	0.059985	0.241717	0.059985	0.193658	0.059985
5	MLPClassifier	0.390947	0.23503	0.390947	0.275803	0.390947	0.238714	0.390947
6	AdaBoostClassifier	0.288868	0.171098	0.288868	0.217414	0.288868	0.181834	0.288868
7	XGBClassifier	0.421955	0.301455	0.421955	0.362506	0.421955	0.288359	0.421955

Model klasifikasi dengan performance tertinggi adalah XGBoostClassifier

# Model Selection (2)

Melatih data training yang telah diresample dengan 8 model klasifikasi

**Menggunakan metode oversampling Synthetic Minority Oversampling Technique (SMOTE)**

	Algorithm	Accuracy	F1 Macro	F1 Micro	Precision Macro	Precision Micro	Recall Macro	Recall Micro
0	LogisticRegression	0.35418	0.249087	0.35418	0.315958	0.35418	0.254724	0.35418
1	KNeighborsClassifier	0.225065	0.134229	0.225065	0.155555	0.225065	0.136123	0.225065
2	DecisionTreeClassifier	0.23184	0.164168	0.23184	0.164387	0.23184	0.164019	0.23184
3	RandomForestClassifier	0.348483	0.200005	0.348483	0.25941	0.348483	0.203592	0.348483
4	GaussianNB	0.06756	0.065598	0.06756	0.216929	0.06756	0.190696	0.06756
5	MLPClassifier	0.37746	0.234458	0.37746	0.310065	0.37746	0.241988	0.37746
6	AdaBoostClassifier	0.204619	0.082176	0.204619	0.108741	0.204619	0.16233	0.204619
7	XGBClassifier	0.420354	0.298803	0.420354	0.354303	0.420354	0.286448	0.420354

**Menggunakan metode undersampling RandomUnderSampling**

	Algorithm	Accuracy	F1 Macro	F1 Micro	Precision Macro	Precision Micro	Recall Macro	Recall Micro
0	LogisticRegression	0.343464	0.244133	0.343464	0.316122	0.343464	0.264833	0.343464
1	KNeighborsClassifier	0.204711	0.132595	0.204711	0.152072	0.204711	0.143882	0.204711
2	DecisionTreeClassifier	0.210192	0.158064	0.210192	0.167139	0.210192	0.172697	0.210192
3	RandomForestClassifier	0.316151	0.198923	0.316151	0.22626	0.316151	0.225244	0.316151
4	GaussianNB	0.172009	0.053469	0.172009	0.209472	0.172009	0.137614	0.172009
5	MLPClassifier	0.346913	0.217424	0.346913	0.275899	0.346913	0.245768	0.346913
6	AdaBoostClassifier	0.254534	0.154987	0.254534	0.182004	0.254534	0.228859	0.254534
7	XGBClassifier	0.371794	0.277307	0.371794	0.28861	0.371794	0.294741	0.371794

**XGBoostClassifier memiliki performance lebih baik dibanding ketujuh model klasifikasi lainnya**

# Model Selection (3)

Melatih model klasifikasi XGBoostClassifier dengan 8 data yang telah diresample dengan metode oversampling dan undersampling

	Algorithm	Accuracy	F1 Macro	F1 Micro	Precision Macro	Precision Micro	Recall Macro	Recall Micro
0	RandomUnderSampler	0.37438	0.28166	0.37438	0.351737	0.37438	0.298317	0.37438
1	NearMiss	0.075597	0.085191	0.075597	0.154913	0.075597	0.212977	0.075597
2	NearMissV2	0.066605	0.081755	0.066605	0.183439	0.066605	0.187319	0.066605
3	NearMissV3	0.164742	0.144705	0.164742	0.16471	0.164742	0.241084	0.164742
4	TomekLinks	0.407667	0.277583	0.407667	0.318386	0.407667	0.269606	0.407667
5	RandomOverSampler	0.416844	0.296851	0.416844	0.351644	0.416844	0.284629	0.416844
6	SMOTE	0.392517	0.281763	0.392517	0.297514	0.392517	0.276201	0.392517
7	BorderlineSMOTE	0.391778	0.282265	0.391778	0.295028	0.391778	0.277371	0.391778

XGBoostClassifier memiliki performance lebih baik dengan menggunakan data yang telah diresample dengan RandomOverSampler

# Model Training

Menggunakan model XGBoostClassifier dengan data yang telah diresample dengan RandomOverSampler

## Detail

```
xgbc = XGBClassifier(  
    learning_rate=0.1,  
    n_estimators=5000,  
    gamma=0.4,  
    subsample=0.8,  
    colsample_bytree=0.8,  
    objective='binary:logistic',  
    nthread=4,  
    seed=27,  
    max_depth=3,  
    min_child_weight=5  
)  
  
oversample = RandomOverSampler(sampling_strategy='minority')  
X_train_over, y_train_over = oversample.fit_resample(X_train, y_train)  
  
xgbc.fit(X_train_over, y_train_over)  
y_pred = xgbc.predict(X_test)  
evaluate_classifier_performance(y_test, y_pred)
```

## Hasil Evaluasi

Accuracy Average: 0.4387682832948422  
F1 Macro Average: 0.33816437869858323  
F1 Micro Average: 0.4387682832948422  
Precision Macro Average: 0.32369466620512244  
Precision Micro Average: 0.4387682832948422  
Recall Macro Average: 0.37459813127580505  
Recall Micro Average: 0.4387682832948422

## Hasil pada Kaggle



classify (16).csv

Complete · Michael Christlambert Si...

0.45125

# Regression



# Catatan Awal

## Target Prediksi

Perlu diperhatikan bahwa model regression ini melakukan prediksi rata-rata ACS dari kedua tim dalam suatu match dalam **level game**, bukan level match ataupun level player.

Hal ini karena

- Performa player dalam suatu tim dependent terhadap performa teammatenya.
- Performa tim dalam game berbeda dan match sama bisa berbeda-beda.
- Rata-rata ACS pada suatu match bisa diturunkan dari rata-rata ACS pada game-game di match tersebut.

Karena fitur-fitur skor yang ada pada dataset hanya menjelaskan skor pada level player, maka diperlukan **feature engineering**.

## Batasan

Terdapat batasan bahwa fitur Kills, Deaths, PlusMinus, dan ADR tidak boleh digunakan demi membuat persoalan ini menjadi lebih menantang dikarenakan korelasinya yang sangat tinggi dengan ACS.



# Pre-Processing

Dataset yang digunakan adalah dataset VPM yang sudah dilakukan preprocessing pada tahap EDA untuk menghilangkan row-row yang tidak valid ditambah dengan:

- **Encoding:** Menggunakan Label Encoding pada fitur **Agent** dan **Map** karena berbentuk kategorikal dan tidak memiliki urutan.
- **Feature Engineering:** Melakukan penurunan fitur-fitur baru dari fitur-fitur yang sudah ada.
  - **agentUseX:** Apakah agent X digunakan oleh tim? Diambil dari fitur Agent demi menjelaskan komposisi dari tim pada suatu permainan.
  - **isWin:** Apakah tim menang? Diambil dari Winner yang bernilai 1 atau 2
  - **AllyMapScore** dan **EnemyMapScore:** Map Score dari tim dan lawannya. Diambil dari **Team1\_MapScore** dan **Team2\_MapScore**
  - Rata-rata **ACS**, **HS\_Percent**, dan skor-skor lainnya dari tim: Diambil dari skor-skor tiap player dalam tim
  - Team **Eco**, **SemiEco**, **SemiBuy**, **Buy**, dan **TotalRounds** dari tim: Diambil dari **Team1\_Eco**, **Team2\_Eco**, dsb.

# Pre-Processing

- **Penanganan Outlier:** Tidak ada outlier yang didrop meskipun ada banyak outlier karena sifatnya yang natural. Sebagai contoh, pada Num\_5Ks, nilai-nilai yang di atas 0 banyak yang dianggap sebagai outlier meskipun tidak jauh. Namun, hal ini karena kejadian dimana suatu pemain membunuh setidaknya 5 musuh dalam satu ronde sangat jarang terjadi.
- **Standarisasi:** Karena beberapa model seperti Linear Regression dan MLP (Neural Network) sensitif terhadap scaling, maka dilakukan standarisasi.
- **Feature Selection:** Setelah melakukan feature selection dengan menghapus fitur-fitur yang tidak relevan dan melakukan **Lasso's Regression** (mempertimbangkan kecepatan komputasinya), didapat fitur-fitur berikut:  
**HS\_Percent, FKFD\_PlusMinus, Num\_2Ks, Num\_3Ks, Num\_4Ks, Num\_5Ks, OnevOne, OnevTwo, OnevThree, Econ, Plants, Defuses, isWin, Team\_Eco, Team\_SemiEco, Team\_SemiBuy, Team\_FullBuy, Team\_TotalRounds, Map, dan Ally\_MapScore**
- **Train-Test Splitting:** Dataset dibagi menjadi 80% untuk training set dan 20% untuk validation set.

# Model Selection (I)

Dilakukan training pada berbagai model regression seperti **Linear Regression**, **Lasso Regression**, **Ridge Regression**, **Random Forest**, dan **MLP** dengan training set. Model-model tersebut dilatih sedemikian sehingga bisa melakukan **prediksi pada rata-rata ACS pemain dari suatu tim dalam suatu game**.

Setelah itu hasil prediksi model-model tersebut pada validation set digunakan untuk mengevaluasi performanya menggunakan metrik-metrik regression dengan menitikberatkan pada **R-squared**.

Berikut adalah hasil yang diperoleh:

Model	MAE	MSE	RMSE	R2
Linear Regression	6.862058	84.848131	9.211305	0.893325
Lasso Regression	6.859216	84.780964	9.207658	0.893410
Ridge Regression	6.859248	84.628469	9.199373	0.893602
Random Forest	6.630926	77.087826	8.779967	0.903082
MLP (NN)	6.104463	63.852451	7.990773	0.919722

Terlihat bahwa model neural network MLP memiliki performa yang paling bagus dalam melakukan prediksi ACS tim, yaitu dengan nilai R-squared **91.9722%**.

# Model Selection (2)

Setelah mengetahui bahwa **MLP memiliki performa terbaik**, dilakukan hyperparameter tuning terhadap model tersebut. Berikut adalah konfigurasi terbaik yang didapat dari hasil tuning:

```
mlp_reg_tuned = MLPRegressor(  
    alpha=0.001,  
    activation='tanh',  
    hidden_layer_sizes=[100],  
    learning_rate='constant',  
    random_state=42  
)  
  
mlp_reg_tuned.fit(x_train, y_train)
```

Model	MAE	MSE	RMSE	R2
Linear Regression	6.862058	84.848131	9.211305	0.893325
Lasso Regression	6.859216	84.780964	9.207658	0.893410
Ridge Regression	6.859248	84.628469	9.199373	0.893602
Random Forest	6.630926	77.087826	8.779967	0.903082
MLP (NN)	6.104463	63.852451	7.990773	0.919722
MLP Hyperparameter Tuned	5.982646	60.524607	7.779756	0.923906

Terdapat kenaikan dari model MLP yang belum dilakukan tuning sebesar **0.484%** dari 91.9722% menjadi **92.3906%**

# Post-Processing

Hasil prediksi dari model MLP tersebut adalah rata-rata ACS dari suatu tim pada suatu game. Untuk mendapatkan rata-rata ACS kedua tim dalam suatu match, dilakukan perhitungan rata-rata ACS game dari tiap tim dalam match.

## Performa Model dalam Kaggle

[Submission and Description](#)

Public Score [\(i\)](#)



[match\\_acs.csv](#)

Complete · Emir Shamsu...

**0.90324**



# Clustering



# Dataset untuk Clustering

## Dataset

Dataset yang digunakan berisikan informasi dari berbagai pertandingan profesional dalam turnamen permainan Valorant. Turnamen yang diambil dari kurun tahun 2020 - 2022 serta memiliki 108250 game.

## Fitur

Dataset yang digunakan adalah hasil preprocessing pada proses sebelumnya yaitu final\_dataset.csv yang memiliki fitur sebagai berikut.

Adapun penjelasan mengenai fitur tersebut terdapat pada bagian sebelumnya.

GameID	int64	Defuses	float64
PlayerName	object	MatchID	int64
TeamAbbreviation	object	Map	object
Agent	object	Winner	int64
ACS	float64	Team1_Eco	float64
Kills	float64	Team1_SemiEco	float64
Deaths	float64	Team1_SemiBuy	float64
Assists	float64	Team1_FullBuy	float64
PlusMinus	float64	Team1_TotalRounds	int64
ADR	float64	Team2_Eco	float64
HS_Percent	float64	Team2_SemiEco	float64
FirstKills	float64	Team2_SemiBuy	float64
FirstDeaths	float64	Team2_FullBuy	float64
FKFD_PlusMinus	float64	Team2_TotalRounds	int64
Num_2Ks	float64	Date	object
Num_3Ks	float64	Patch	float64
Num_4Ks	float64	EventID	int64
Num_5Ks	float64	EventName	object
OnevOne	float64	EventStage	object
OnevTwo	float64	Team1ID	int64
OnevThree	float64	Team2ID	int64
OnevFour	float64	Team1	object
OnevFive	float64	Team2	object
Econ	float64	Team1_MapScore	int64
Plants	float64	Team2_MapScore	int64

# Target Clustering

## Untuk Pemain Valorant

Pemain dapat menggunakan hasil clustering ini sebagai pertimbangan dalam memilih Agent apa yang akan digunakan dan Map apa yang cocok dengan Agent tersebut agar mendapatkan nilai ACS tertinggi.

## Untuk Pengembang Permainan

Data hasil clustering ini dapat digunakan sebagai pertimbangan untuk melakukan balancing dalam Agent sehingga setiap Agent diharapkan memiliki kelebihan dan kelemahan yang setara. Selain itu juga dapat digunakan sebagai pertimbangan untuk melakukan redesign Map sehingga tidak terjadi kecenderungan pemain akan memilih Map tertentu karena dianggap lebih mudah.

# Persiapan sebelum Clustering

## Clustering Untuk Nilai ACS berdasarkan Map dan Agent

Fitur yang akan di-cluster adalah fitur yang berhubungan dengan nilai ACS, yaitu fitur: 'ACS', 'Kills', 'Deaths', 'Assists', 'ADR', dan 'FirstKills'. Serta fitur 'Agent' dan 'Map' untuk melihat bagaimana keterhubungan nilai ACS dengan Map dan Agent yang digunakan.

## Preprocessing Data

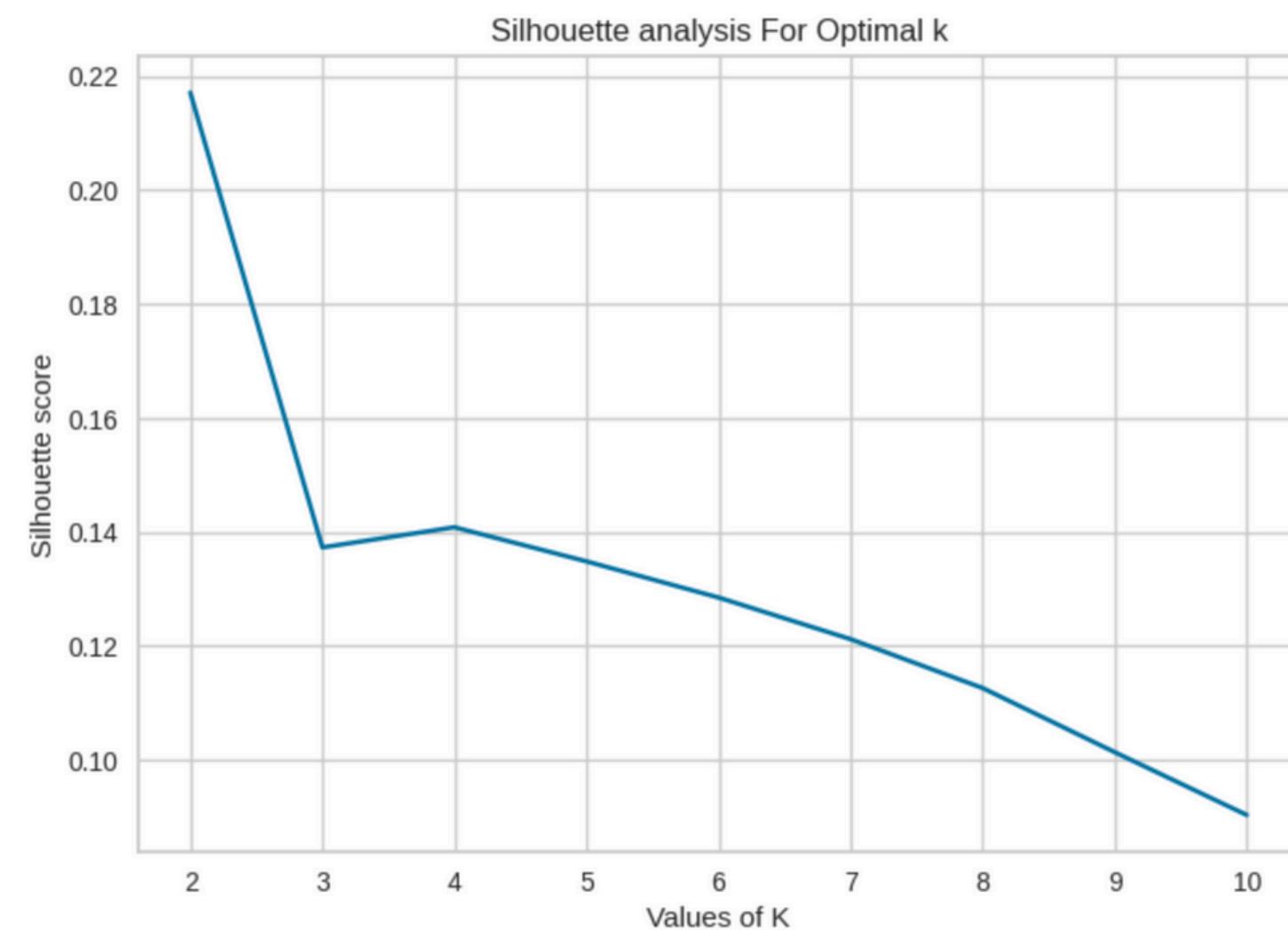
Sebelum data dapat digunakan sebagai masukan pada model clustering, perlu dilakukan preprocessing data agar model dapat lebih akurat dalam melakukan clustering. Berikut ringkasan apa saja preprocessing data yang dilakukan:

- Buang fitur yang tidak digunakan sehingga menyisakan fitur yang dibutuhkan saja
- Lakukan one-hot encoding pada fitur 'Agent' dan 'Map' karena berbentuk kategorikal dan tidak memiliki urutan
- Lakukan standarisasi (menggunakan StandarScaler) pada fitur yang lainnya

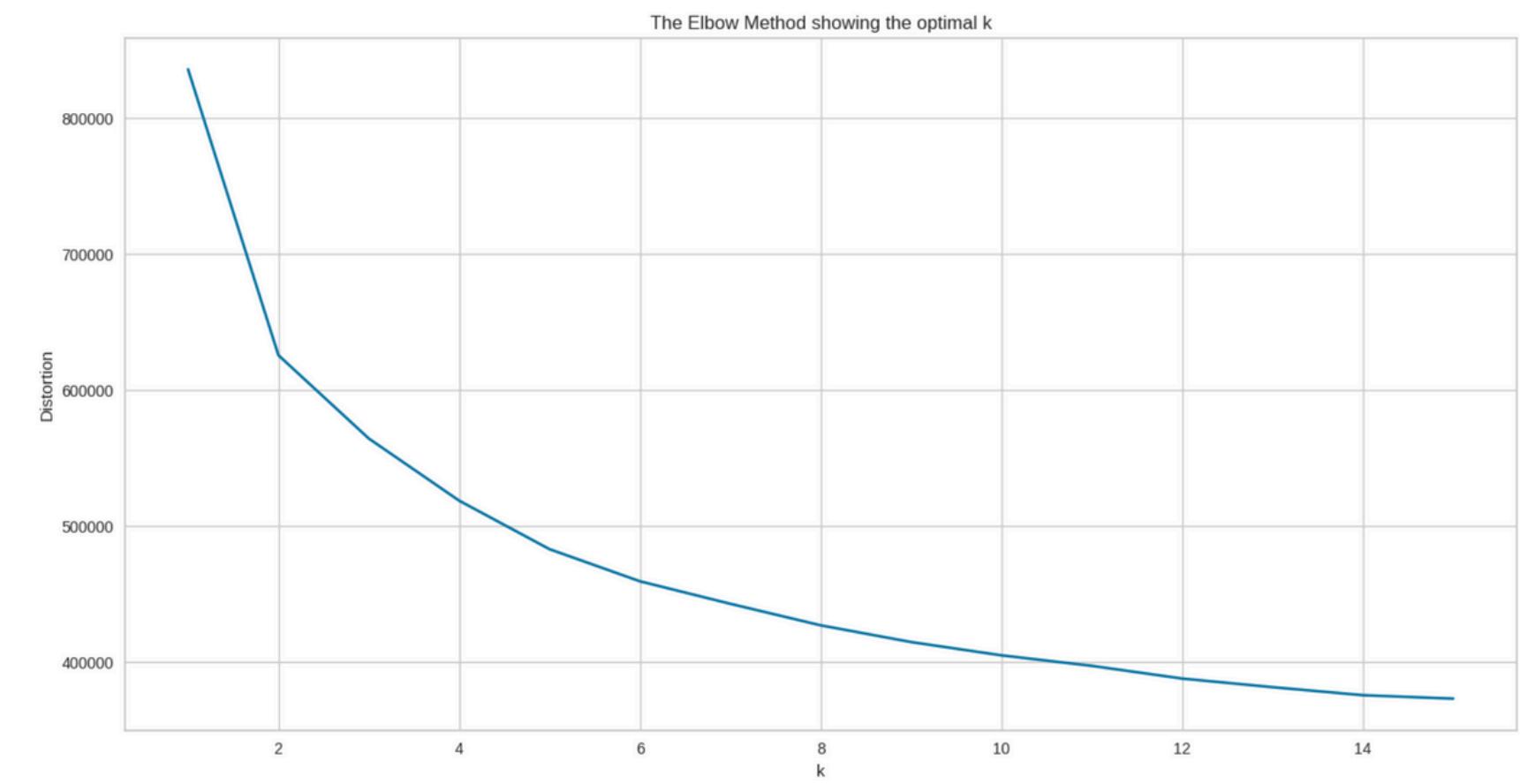
# Persiapan Data

Setelah dilakukan preprocessing data, ada beberapa tahapan lagi sebelum dapat melakukan klasterisasi, yaitu mencari jumlah klaster terbaik menggunakan Elbow Method dan Silhouette Score. Berdasarkan hasil keduanya, jumlah klaster terbaik yang bisa didapatkan adalah 4.

## Silhouette Score



## Elbow Method



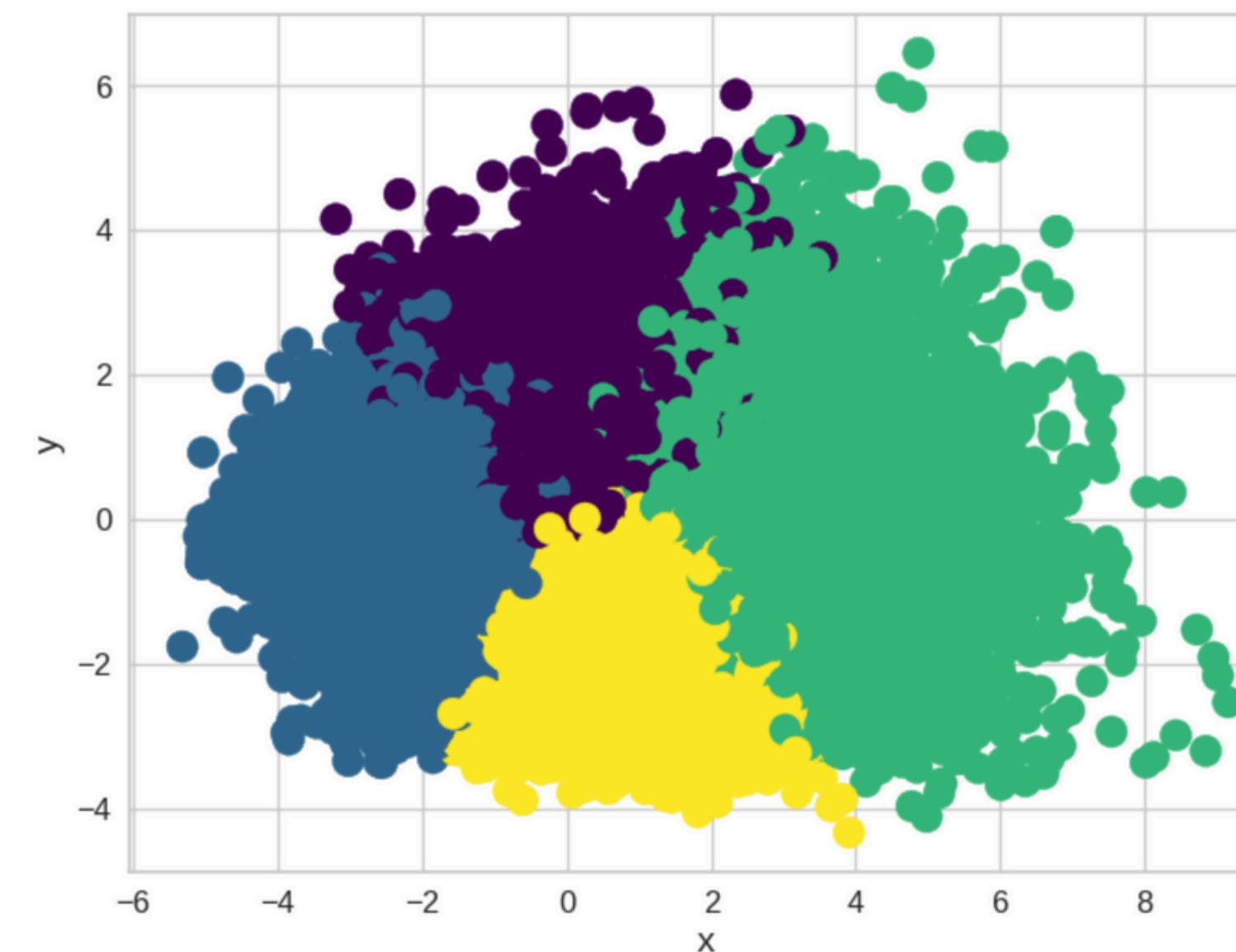
# Hasil Clustering



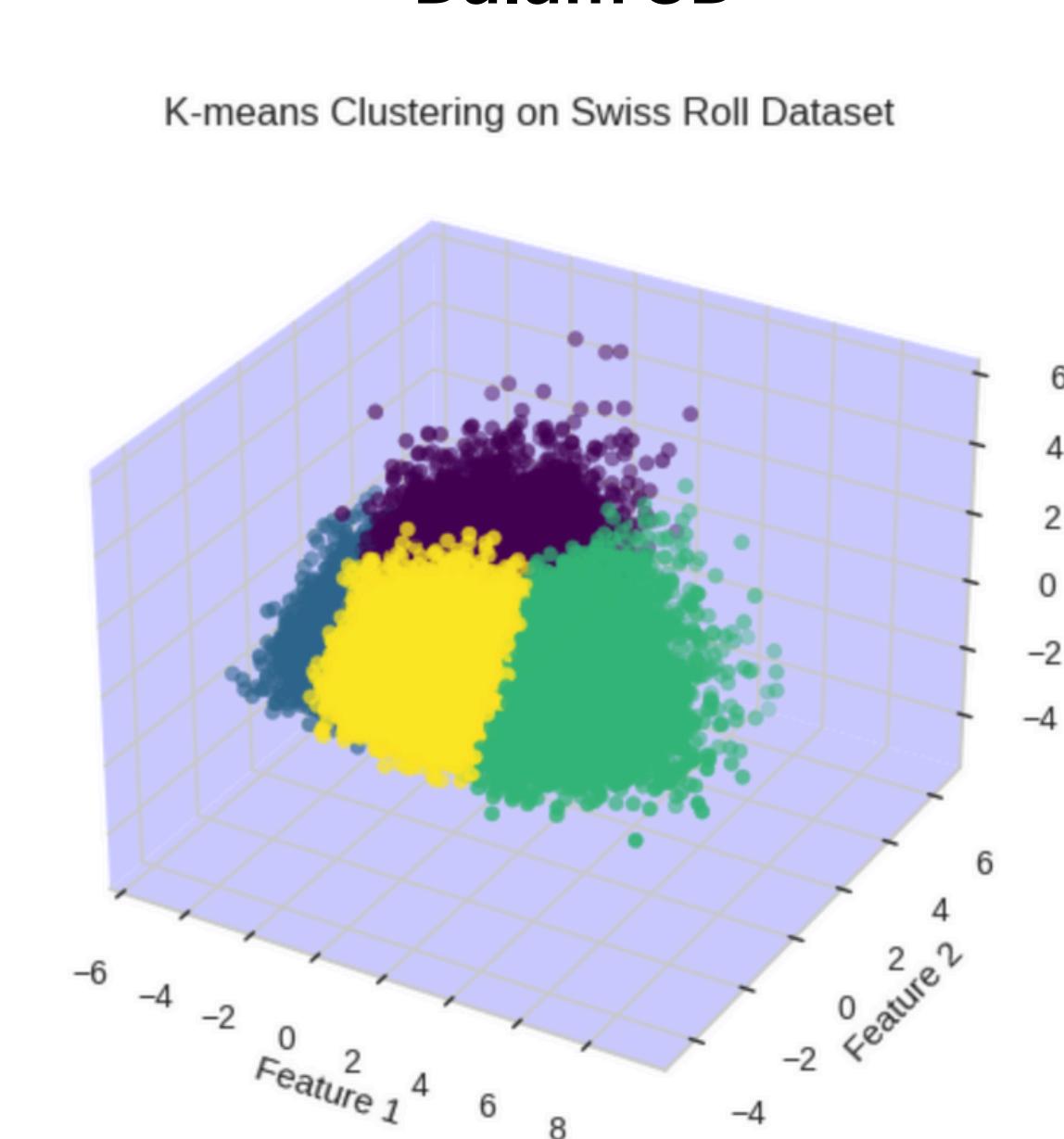
# Visualisasi Data Hasil Clustering

Karena jumlah fitur lebih dari 2, maka visualisasi dapat menggunakan reduksi PCA agar bisa menjadi 2 dan 3 dimensi sehingga lebih mudah dilihat.

Dalam 2D

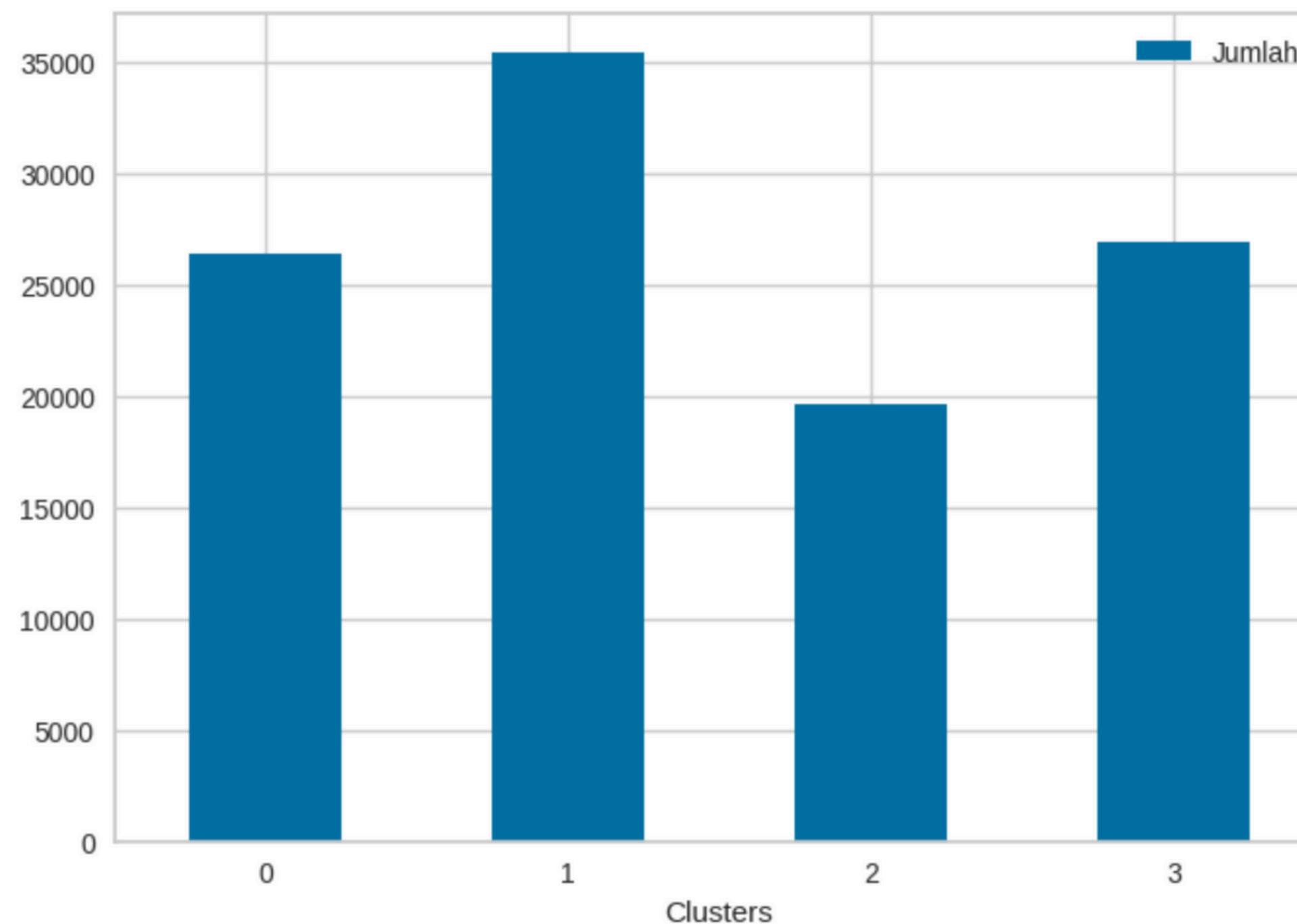


Dalam 3D



# Data Hasil Clustering (I)

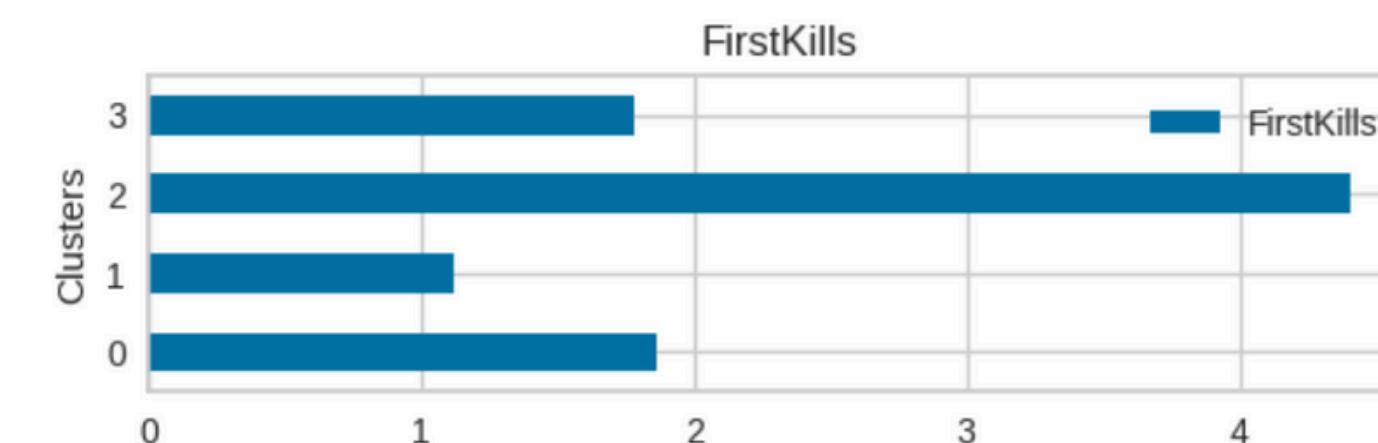
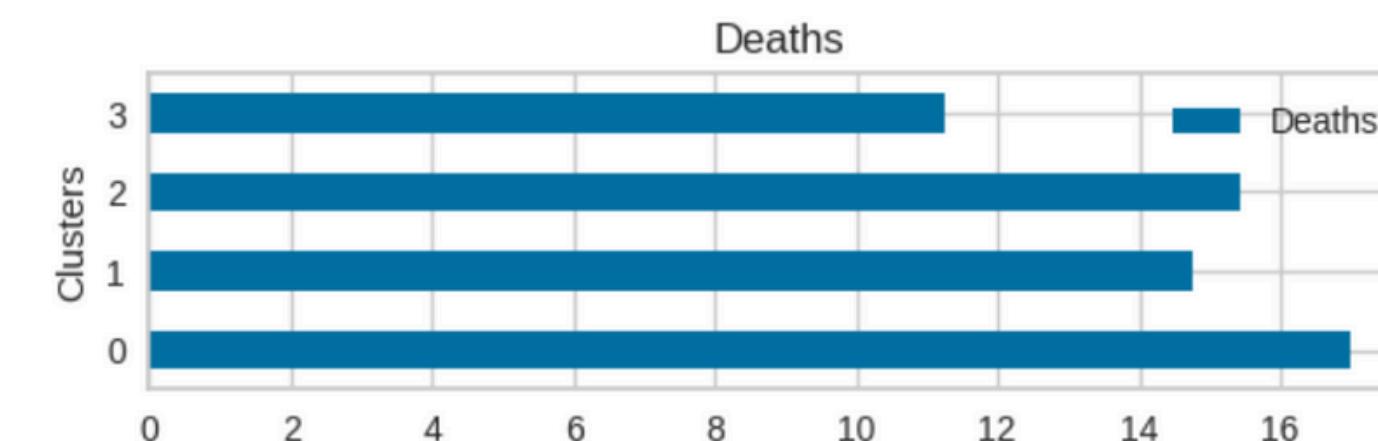
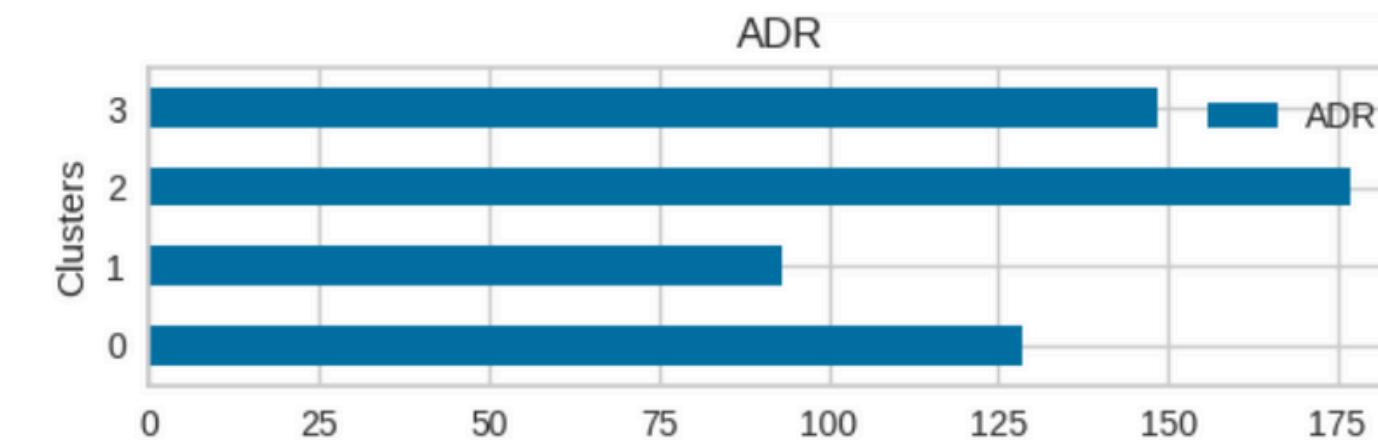
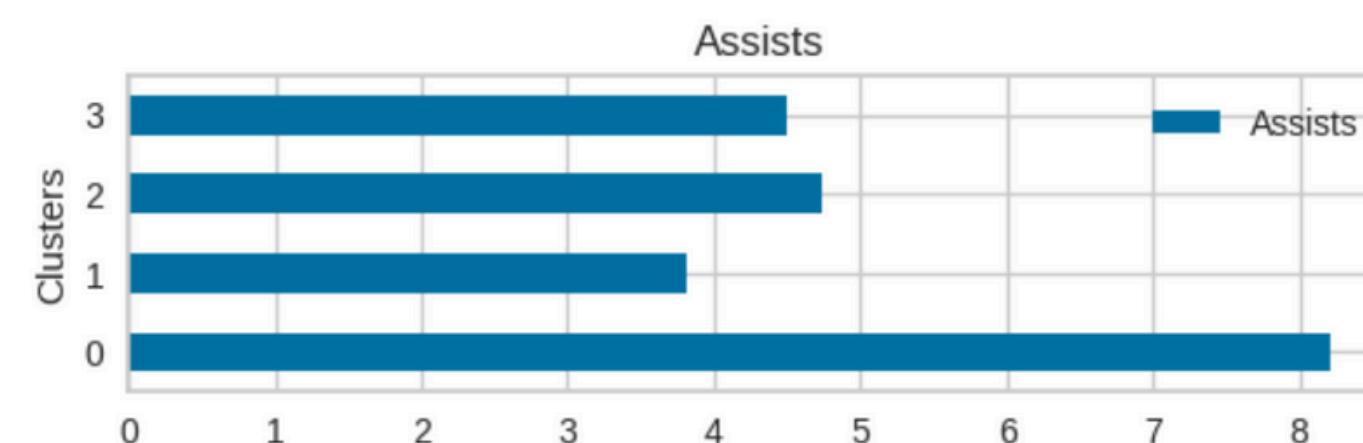
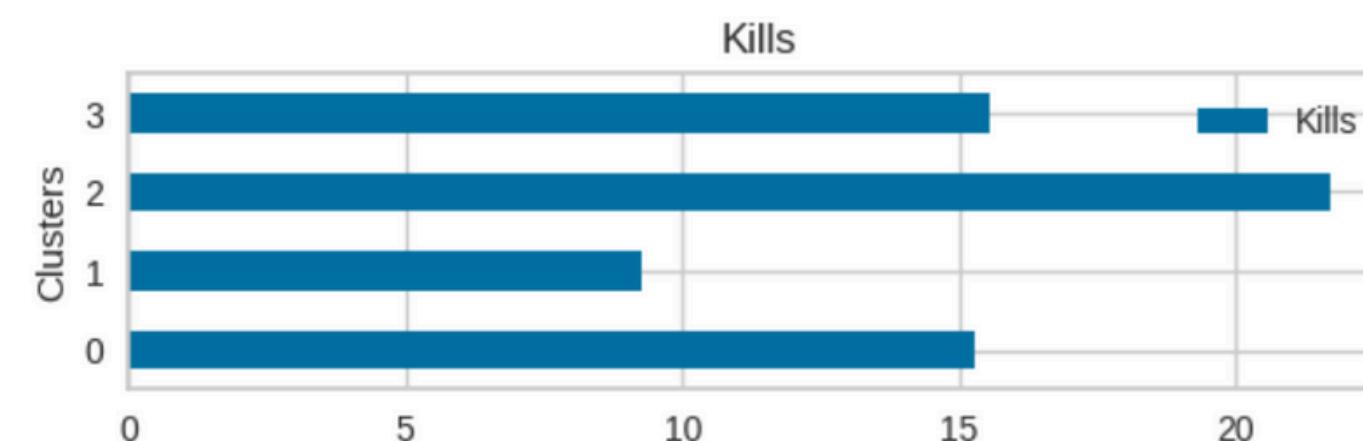
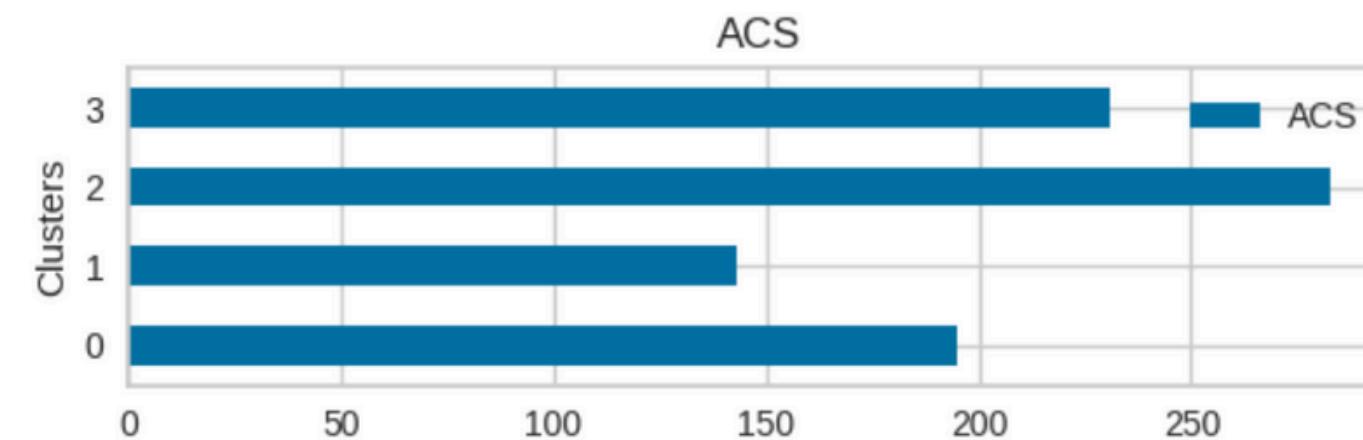
## Jumlah Data Tiap Cluster



Berdasarkan data, Cluster 1 memiliki data terbanyak sedangkan Cluster 2 memiliki data terkecil

# Data Hasil Clustering (2-I)

## Rerata Fitur Per Cluster



# Data Hasil Clustering (2-2)

## Rerata Fitur Per Cluster

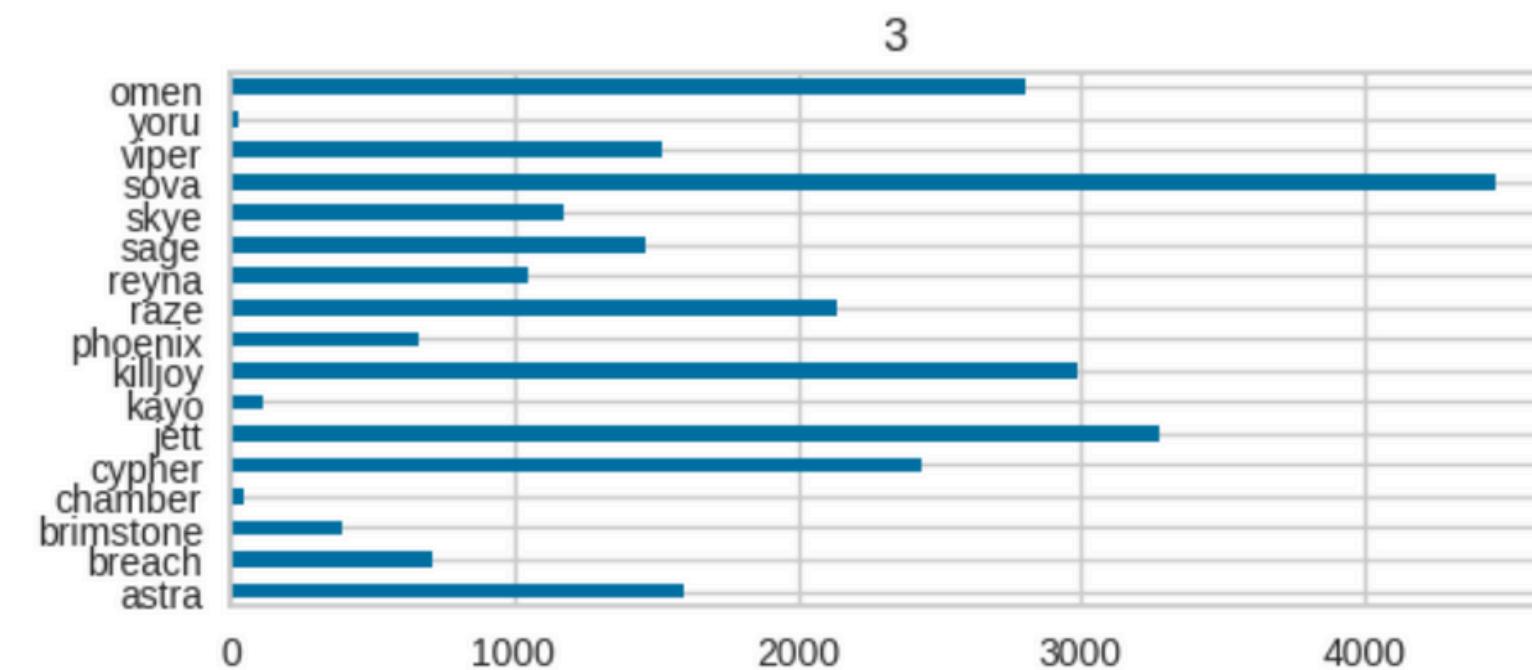
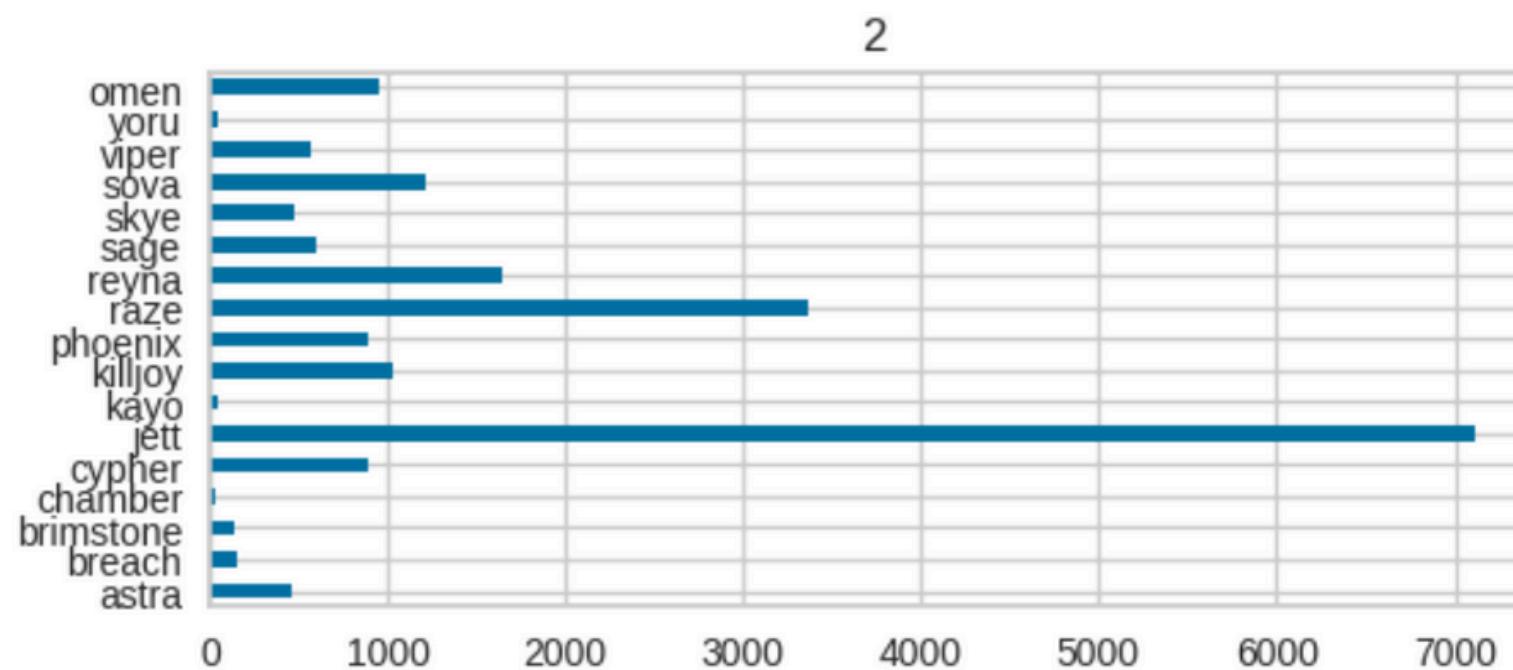
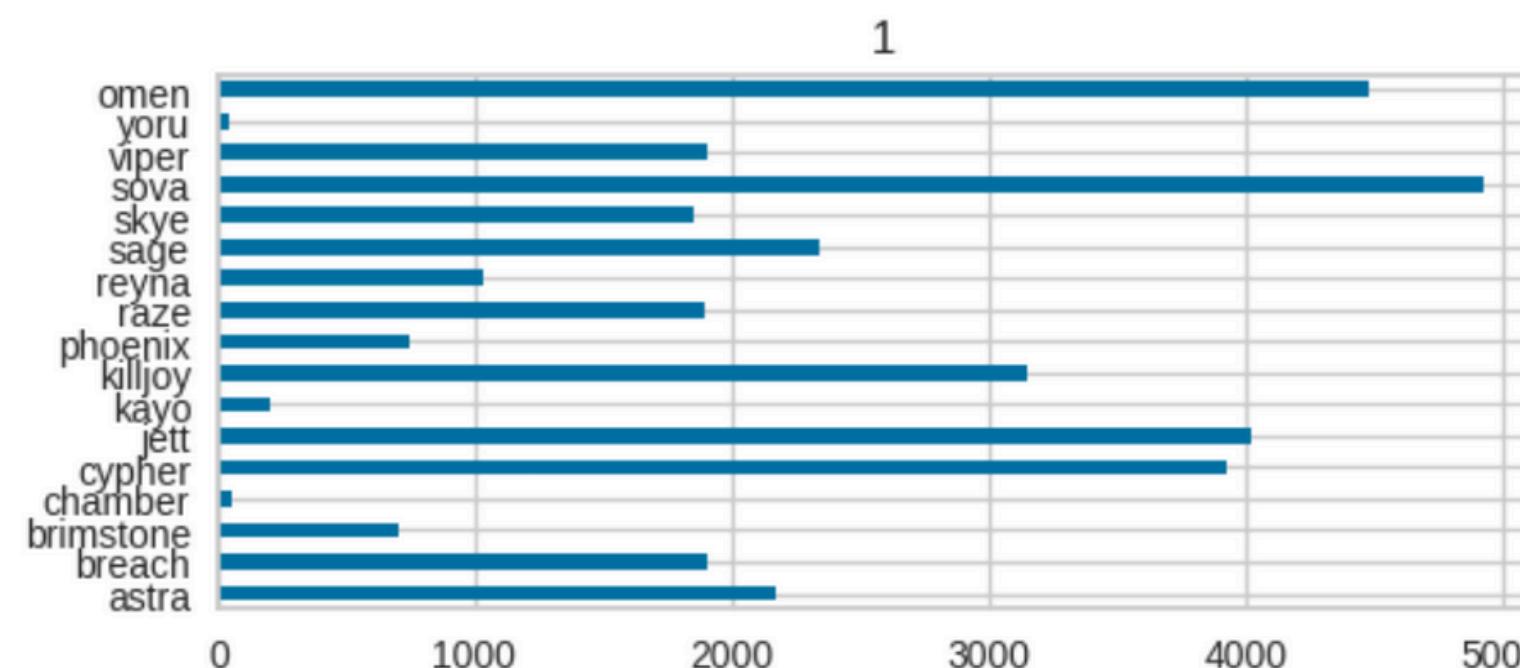
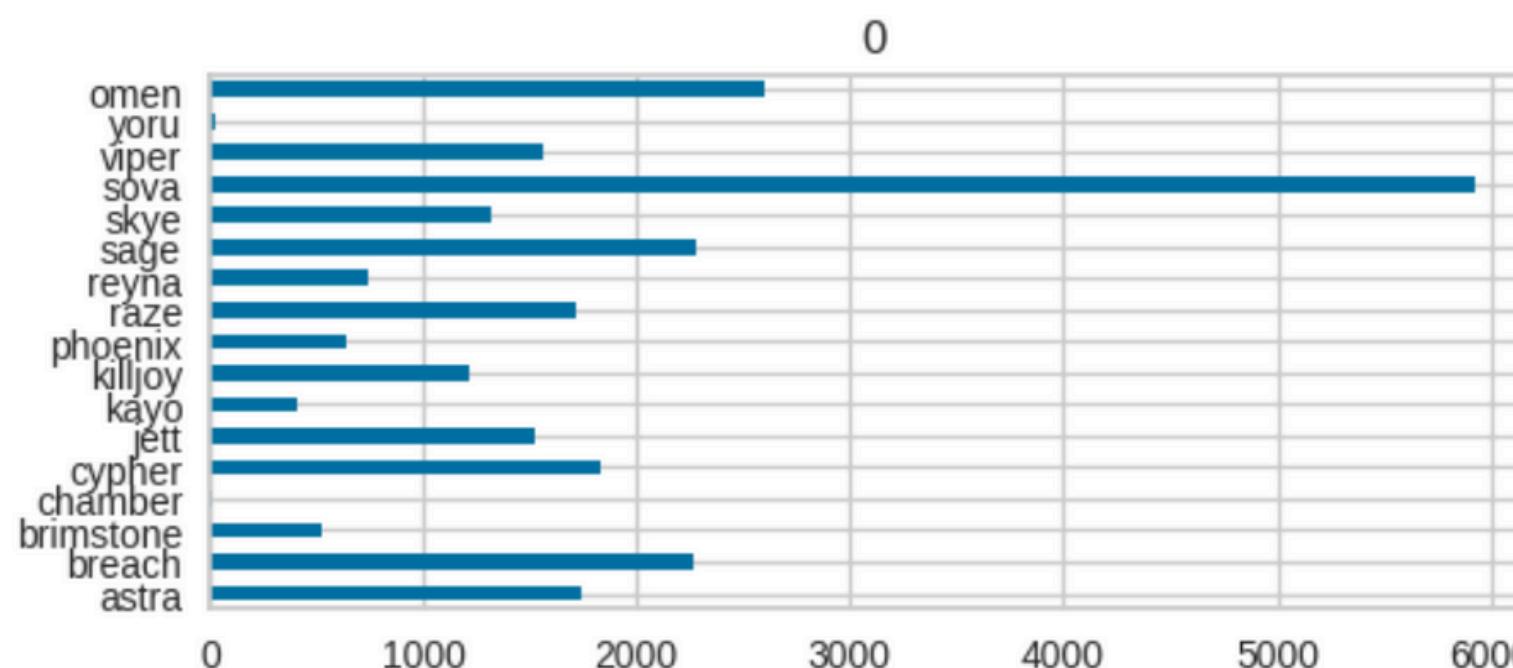
Clusters	ACS	Kills	Deaths	Assists	ADR	FirstKills
0	194.825960	15.292192	16.996813	8.209098	128.397860	1.860601
1	142.796801	9.255588	14.752889	3.820453	93.311707	1.118059
2	282.596209	21.694863	15.446137	4.732572	176.906390	4.409600
3	230.885830	15.551691	11.252818	4.499461	148.488300	1.778133

Terlihat bahwa Cluster 2 memiliki rerata nilai ACS tertinggi, hal ini sesuai dengan rumus perhitungannya yang berbanding lurus dengan jumlah Kills, Assists, ADR, dan FirstKills. Sedangkan, Cluster 1 memiliki rerata nilai ACS terkecil, sesuai dengan rerata nilai lainnya yang juga memiliki nilai terkecil.

Urutan rerata nilai ACS adalah sebagai berikut (dari terbesar ke terkecil):  
Cluster 2 > Cluster 3 > Cluster 0 > Cluster 1

# Data Hasil Clustering (3-I)

## Agent yang digunakan per Cluster



# Data Hasil Clustering (3-2)

## Agent yang digunakan per Cluster

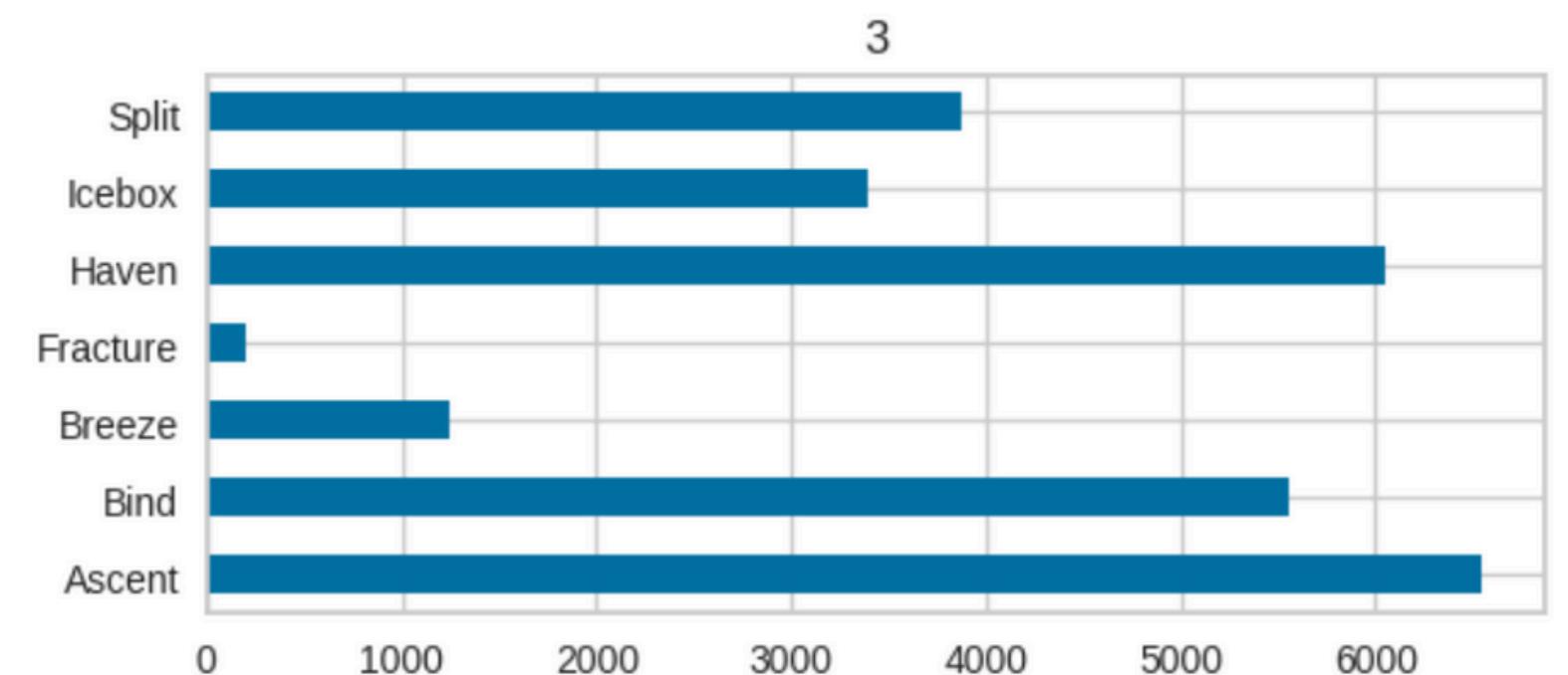
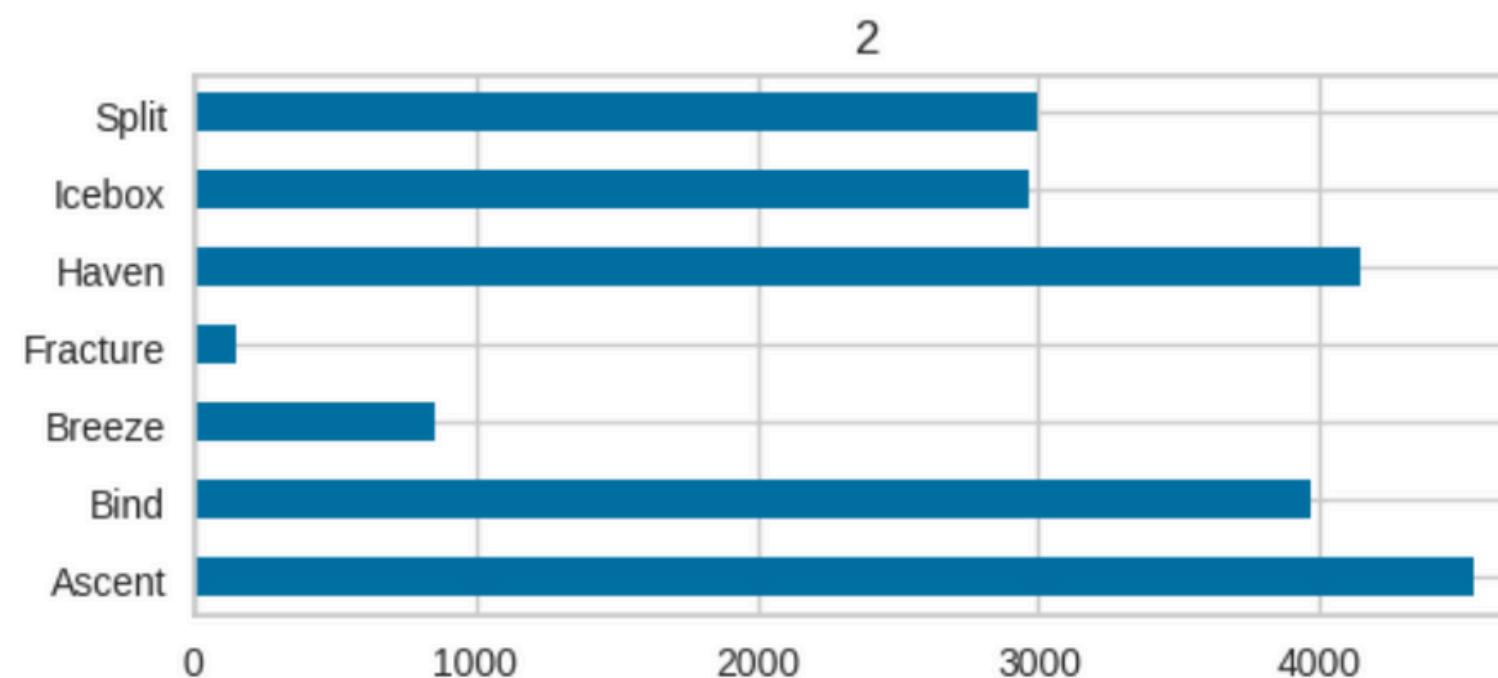
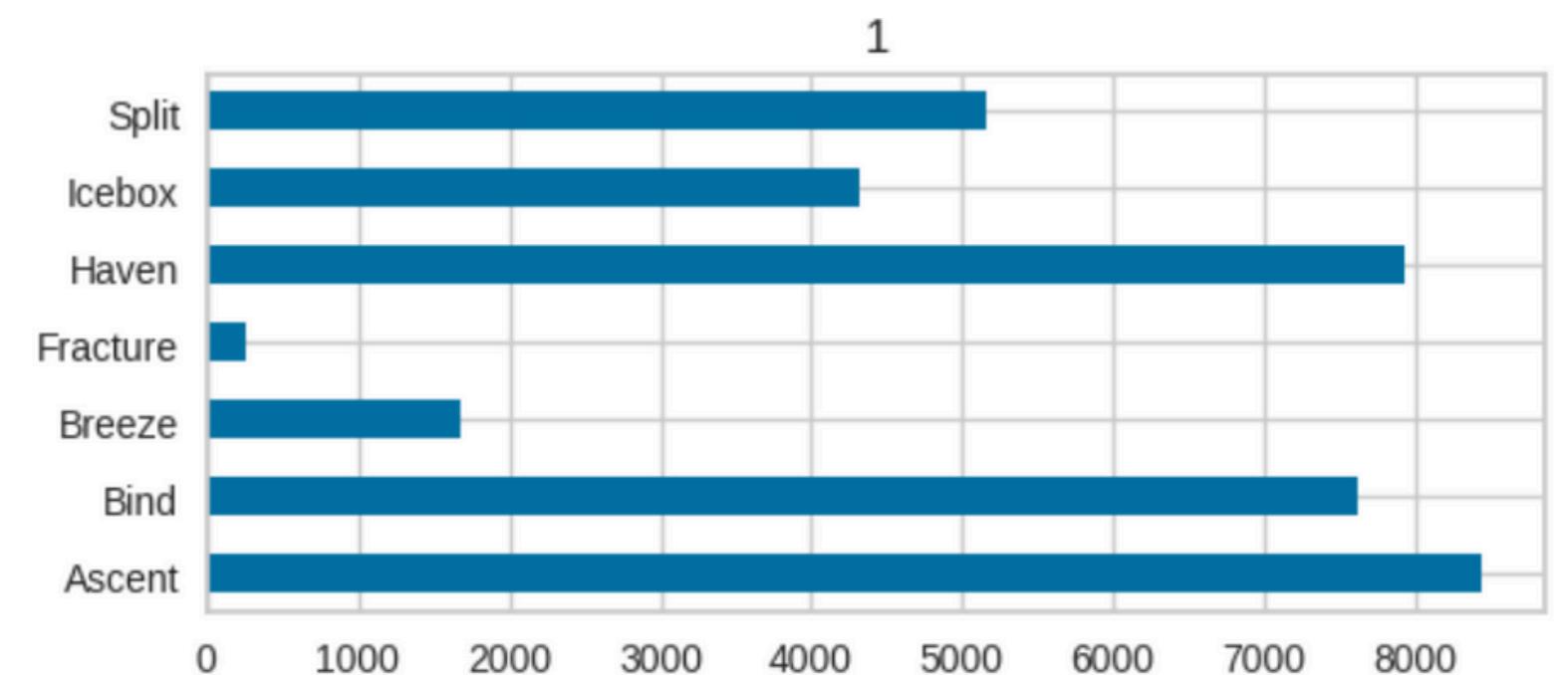
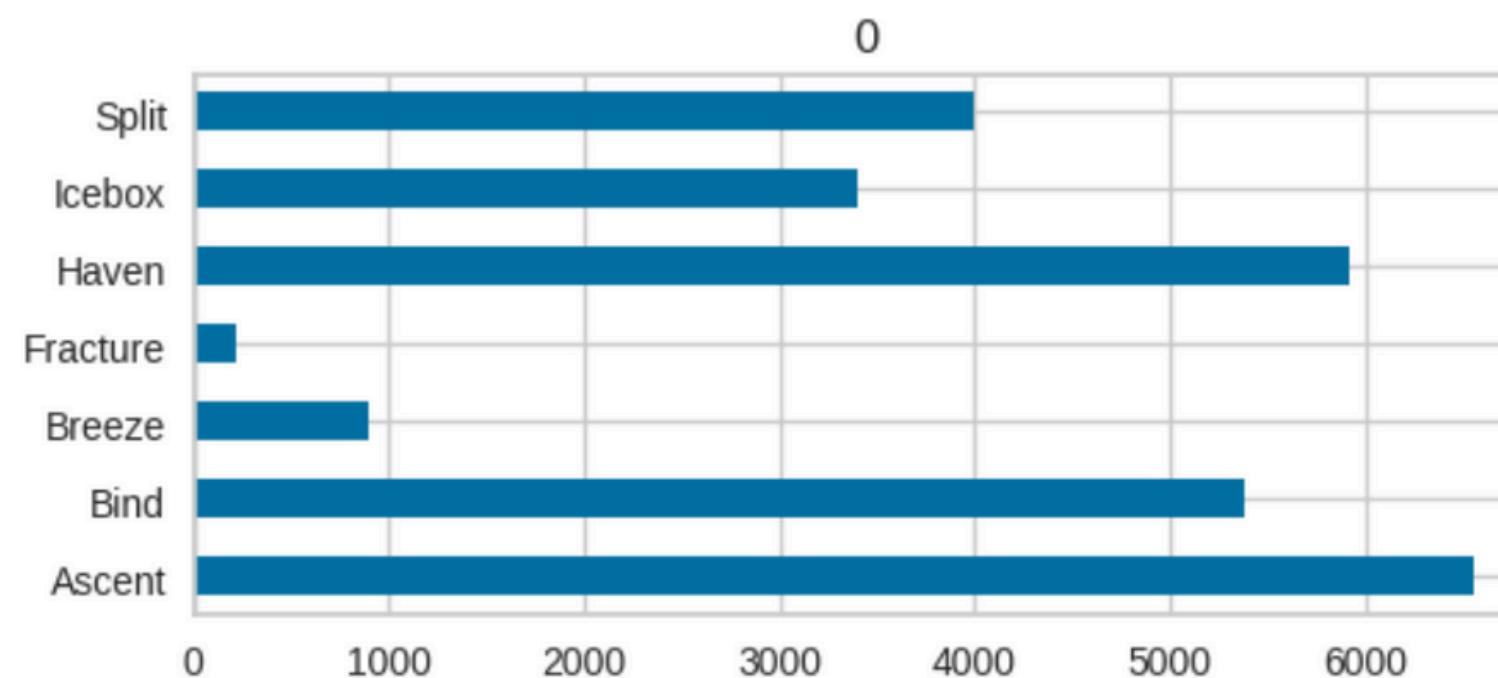
	name	astra	breach	brimstone	chamber	cypher	jett	kayo	killjoy	phoenix	raze	reyna	sage	skye	sova	viper	yoru	omen
Cluster 0		1745	2270	522	14	1832	1521	412	1219	641	1721	746	2283	1313	5929	1557	31	2600
Cluster 1		2173	1905	704	53	3930	4026	203	3153	749	1900	1029	2342	1854	4932	1907	42	4487
Cluster 2		454	161	144	31	886	7117	53	1027	884	3362	1642	599	473	1216	575	45	955
Cluster 3		1602	717	391	53	2441	3272	118	2983	664	2135	1052	1460	1172	4462	1525	31	2803

Secara ringkas, Agent per Cluster yang dominan digunakan adalah:

- Cluster 0: Sova
- Cluster 1: Sova, Omen
- Cluster 2: Jett
- Cluster 3: Sova

# Data Hasil Clustering (4-1)

## Jumlah Map yang digunakan per Cluster



# Data Hasil Clustering (4-2)

Jumlah Map yang digunakan per Cluster

	<b>name</b>	<b>Ascent</b>	<b>Bind</b>	<b>Breeze</b>	<b>Fracture</b>	<b>Haven</b>	<b>Icebox</b>	<b>Split</b>
Cluster 0		6558	5379	893	208	5918	3401	3999
Cluster 1		8437	7624	1673	260	7921	4325	5149
Cluster 2		4547	3968	854	147	4143	2970	2995
Cluster 3		6548	5559	1240	205	6058	3394	3877

Dapat terlihat bahwa Map yang digunakan untuk masing-masing cluster memiliki proporsi jumlah yang hampir sama, yaitu (dari besar ke kecil):

Ascent > Haven > Bind > Split > Icebox > Breeze > Fracture

# Analisis Hasil Clustering



# Interpretasi Cluster (I)

Berdasarkan hasil clustering, masing-masing Cluster dapat diinterpretasikan sebagai berikut.

Cluster 0	Cluster 1	Cluster 2	Cluster 3
<ul style="list-style-type: none"><li>• Rata-rata nilai ACS sebesar 194.825960</li><li>• Rata-rata nilai Kills sebesar 15.292192</li><li>• Rata-rata nilai Deaths sebesar 16.996813</li><li>• Rata-rata nilai Assists sebesar 8.209098</li><li>• Rata-rata nilai ADR sebesar 128.397860</li><li>• Rata-rata nilai FirstKills sebesar 1.860601</li><li>• Agent yang digunakan sebagai besar adalah Sova</li></ul>	<ul style="list-style-type: none"><li>• Rata-rata nilai ACS sebesar 142.796801</li><li>• Rata-rata nilai Kills sebesar 9.255588</li><li>• Rata-rata nilai Deaths sebesar 14.752889</li><li>• Rata-rata nilai Assists sebesar 3.820453</li><li>• Rata-rata nilai ADR sebesar 93.311707</li><li>• Rata-rata nilai FirstKills sebesar 1.118059</li><li>• Agent yang digunakan sebagian besar adalah Sova dan Omen</li></ul>	<ul style="list-style-type: none"><li>• Rata-rata nilai ACS sebesar 282.596209</li><li>• Rata-rata nilai Kills sebesar 21.694863</li><li>• Rata-rata nilai Deaths sebesar 15.446137</li><li>• Rata-rata nilai Assists sebesar 4.732572</li><li>• Rata-rata nilai ADR sebesar 176.906390</li><li>• Rata-rata nilai FirstKills sebesar 4.409600</li><li>• Agent yang digunakan sebagian besar adalah Jett</li></ul>	<ul style="list-style-type: none"><li>• Rata-rata nilai ACS sebesar 230.885830</li><li>• Rata-rata nilai Kills sebesar 15.551691</li><li>• Rata-rata nilai Deaths sebesar 11.252818</li><li>• Rata-rata nilai Assists sebesar 4.499461</li><li>• Rata-rata nilai ADR sebesar 148.488300</li><li>• Rata-rata nilai FirstKills sebesar 1.778133</li><li>• Agent yang digunakan sebagian besar adalah Sova</li></ul>

Setiap cluster memiliki kesamaan dalam hal Map yang digunakan yaitu: Ascent > Haven > Bind > Split > Icebox > Breeze > Fracture (dari besar ke kecil)

# Interpretasi Cluster (2)

Beberapa analisis mengenai Agent yang dapat diperoleh dari hasil clustering adalah sebagai berikut:

- Agent Jett, seperti pada Cluster 2, cenderung bisa mendapatkan nilai ACS yang tinggi dibandingkan dengan Agent lainnya
- Agent Jett lebih sering mendapatkan FirstKills dibandingkan Agent lain
- Agent Jett lebih banyak memperoleh nilai ADR dan Kills dibandingkan Agent lain, tetapi Deaths yang diperoleh juga tinggi
- Agent Sova cukup sering mendapatkan Deaths, tetapi memberikan Assists yang paling tinggi
- Agent Sova juga memiliki ADR yang cukup tinggi, tetapi lebih kecil dari Agent Jett
- Agent Omen memperoleh nilai ADR dan jumlah Kills yang cukup kecil

Beberapa analisis mengenai Map yang dapat diperoleh dari hasil clustering adalah sebagai berikut:

- Map Ascent, Haven, dan Bind cukup sering digunakan oleh Pemain
- Map Split dan Icebox memiliki tingkat pemilihan yang hampir serupa, meskipun tidak begitu sering digunakan
- Map Breeze sudah cukup jarang digunakan, meskipun masih lebih dari 4x lipat lebih banyak dari Map Fracture
- Map Fracture adalah map yang paling jarang digunakan, bahkan pada semua Cluster yang ada

# Referensi



# Referensi

- <https://www.thegamer.com/valorant-best-duelist-agents-ranked/>
- <https://thesportsrush.com/valorant-initiator-tier-list/>
- [https://valorant.fandom.com/wiki/VALORANT\\_Wiki](https://valorant.fandom.com/wiki/VALORANT_Wiki)
- <https://www.youtube.com/@proguidesvalorant/>
- <https://whatifgaming.com/how-to-calculate-combat-score-in-valorant/>

# Sumber Gambar

- <https://www.behance.net/gallery/127898543/Free-VALORANT-Renders>

**THANK YOU**

