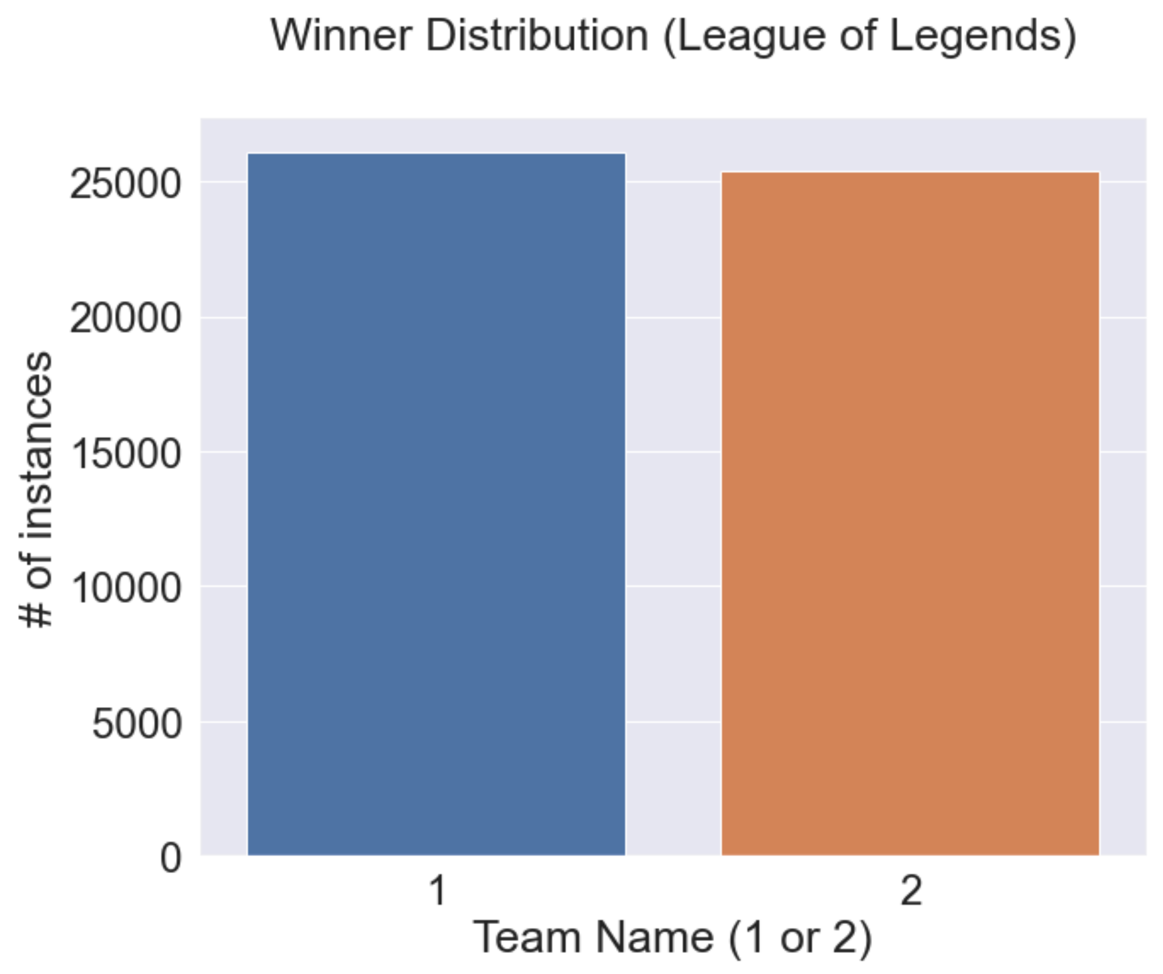
SECTION 2 PROJECT (S2P)

Machine Learning : **From Iron, Diamond, to Challenger**

Analyzing game pattern in League of Legends

AI 05 도현진 (Hannah Do)

0. EDA



Class Distribution (Winning Team)

League of Legends Ranked Games

Mitchell J / Kaggle (2017)

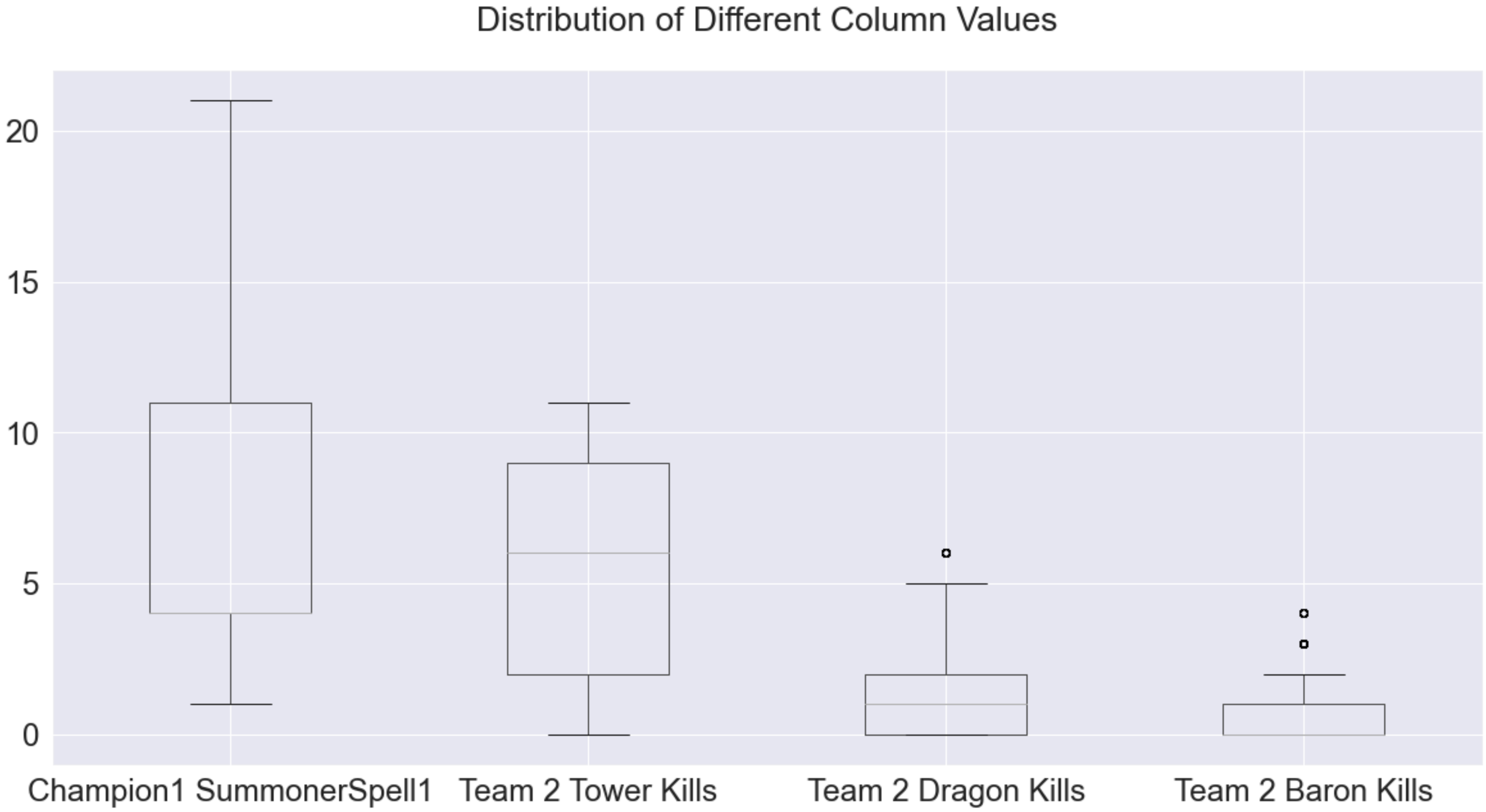
Collection of over 50,000 ranked EUW games from the game League of Legends, as well as json files containing a way to convert between champion and summoner spell IDs and their names. For each game, there are fields for:

- Game ID
- Creation Time (in Epoch format)
- Game Duration (in seconds)
- Season ID
- Winner (1 = team1, 2 = team2)
- First Baron, dragon, tower, blood, inhibitor and Rift Herald (1 = team1, 2 = team2, 0 = none)
- Champions and summoner spells for each team (Stored as Riot's champion and summoner spell IDs)
- The number of tower, inhibitor, Baron, dragon and Rift Herald kills each team has
- The 5 bans of each team (Again, champion IDs are used)

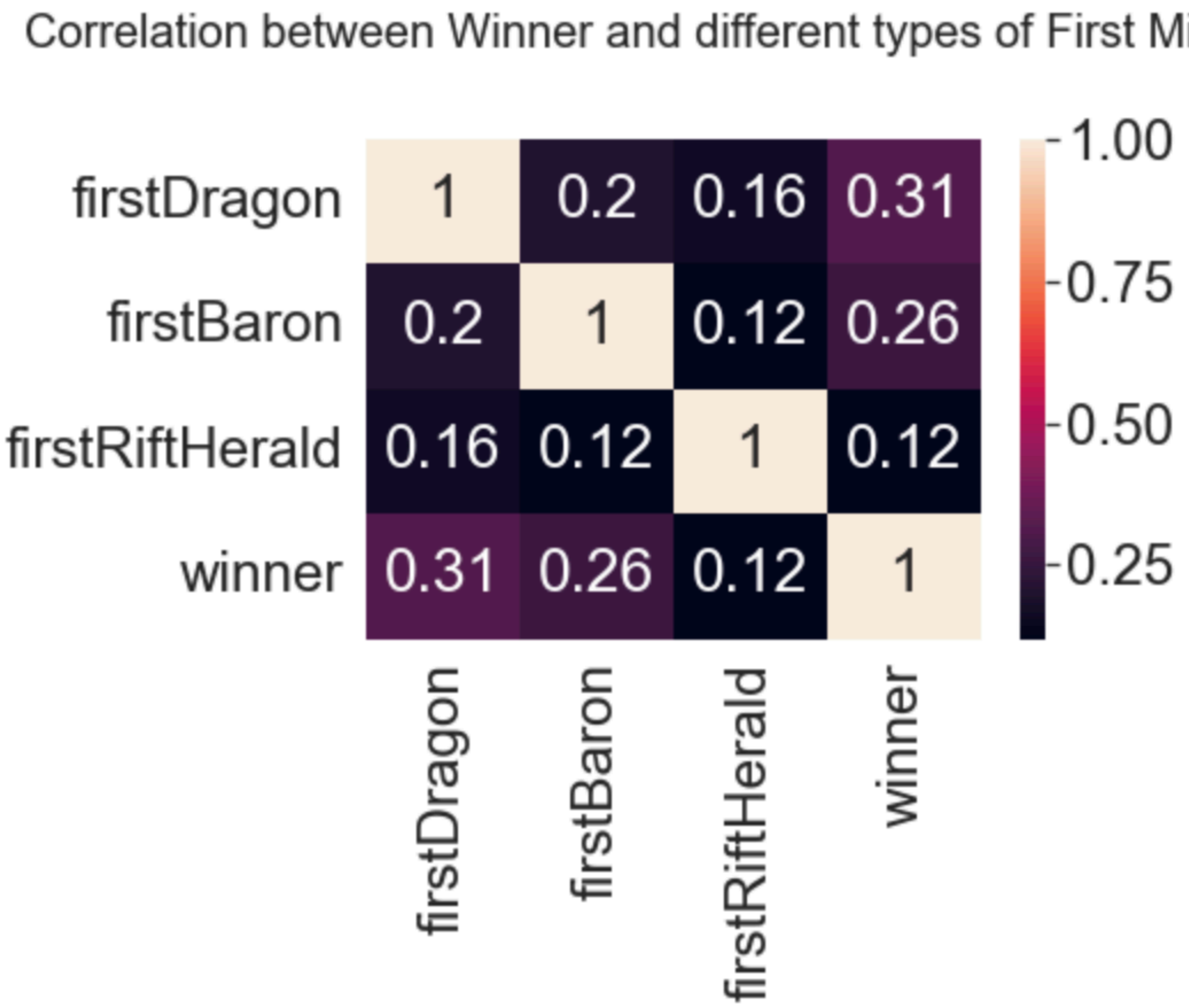
https://www.kaggle.com/datasnaek/league-of-legends?select=games.csv

0. EDA

Summoner spells per each champion,
Number of towers,
inhibitors,
and baron kills per each team
(Distribution shown as a box plot)



0. EDA



According to the heatmap,
***First Dragon and Winner** showed the*
highest correlation score of 0.31

0. EDA

```
champ_info1.head()
```

	type	version	data
Aatrox	champion	7.18.1	{'tags': ['Fighter', 'Tank'], 'title': 'the Da...
Ahri	champion	7.18.1	{'tags': ['Mage', 'Assassin'], 'title': 'the N...
Akali	champion	7.18.1	{'tags': ['Assassin'], 'title': 'the Fist of S...
Alistar	champion	7.18.1	{'tags': ['Tank', 'Support'], 'title': 'the Mi...
Amumu	champion	7.18.1	{'tags': ['Tank', 'Mage'], 'title': 'the Sad M...

```
champ_info1['data'][0]
```

```
{'tags': ['Fighter', 'Tank'],
 'title': 'the Darkin Blade',
 'id': 266,
 'key': 'Aatrox',
 'name': 'Aatrox'}
```

```
champ_info2.head()
```

	type	version	data
1	champion	7.17.2	{'title': 'the Dark Child', 'id': 1, 'key': 'A...
10	champion	7.17.2	{'title': 'The Judicator', 'id': 10, 'key': 'K...
101	champion	7.17.2	{'title': 'the Magus Ascendant', 'id': 101, 'k...
102	champion	7.17.2	{'title': 'the Half-Dragon', 'id': 102, 'key':...
103	champion	7.17.2	{'title': 'the Nine-Tailed Fox', 'id': 103, 'k...

```
champ_info2['data'][266]
```

```
{'title': 'the Darkin Blade', 'id': 266, 'key': 'Aatrox',
'name': 'Aatrox'}
```

Additional json files for champion ID
and detailed champion information in 'data' column

1. Feature Engineering

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51490 entries, 0 to 51489
Data columns (total 60 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   gameDuration        51490 non-null  int64
 1   winner              51490 non-null  int64
 2   firstBlood          51490 non-null  int64
 3   firstTower          51490 non-null  int64
 4   firstInhibitor      51490 non-null  int64
 5   firstBaron          51490 non-null  int64
 6   firstDragon         51490 non-null  int64
 7   firstRiftHerald     51490 non-null  int64
 8   t1_champ1id         51490 non-null  int64
 9   t1_champ1_sum1      51490 non-null  int64
 10  t1_champ1_sum2      51490 non-null  int64
 11  t1_champ2id         51490 non-null  int64
 12  t1_champ2_sum1      51490 non-null  int64
 13  t1_champ2_sum2      51490 non-null  int64
 14  t1_champ3id         51490 non-null  int64
 15  t1_champ3_sum1      51490 non-null  int64
 16  t1_champ3_sum2      51490 non-null  int64
 17  t1_champ4id         51490 non-null  int64
 18  t1_champ4_sum1      51490 non-null  int64
 19  t1_champ4_sum2      51490 non-null  int64
 20  t1_champ5id         51490 non-null  int64
 21  t1_champ5_sum1      51490 non-null  int64
 22  t1_champ5_sum2      51490 non-null  int64
 23  t1_towerKills       51490 non-null  int64
 24  t1_inhibitorKills   51490 non-null  int64
 25  t1_baronKills       51490 non-null  int64
 26  t1_dragonKills      51490 non-null  int64
 27  t1_riftHeraldKills  51490 non-null  int64
 28  t2_champ1id         51490 non-null  int64
 29  t2_champ1_sum1      51490 non-null  int64
 30  t2_champ1_sum2      51490 non-null  int64
```

```
 30  t2_champ1_sum2      51490 non-null  int64
 31  t2_champ2id         51490 non-null  int64
 32  t2_champ2_sum1      51490 non-null  int64
 33  t2_champ2_sum2      51490 non-null  int64
 34  t2_champ3id         51490 non-null  int64
 35  t2_champ3_sum1      51490 non-null  int64
 36  t2_champ3_sum2      51490 non-null  int64
 37  t2_champ4id         51490 non-null  int64
 38  t2_champ4_sum1      51490 non-null  int64
 39  t2_champ4_sum2      51490 non-null  int64
 40  t2_champ5id         51490 non-null  int64
 41  t2_champ5_sum1      51490 non-null  int64
 42  t2_champ5_sum2      51490 non-null  int64
 43  t2_towerKills       51490 non-null  int64
 44  t2_inhibitorKills   51490 non-null  int64
 45  t2_baronKills       51490 non-null  int64
 46  t2_dragonKills      51490 non-null  int64
 47  t2_riftHeraldKills  51490 non-null  int64
 48  t1_tank             51490 non-null  int64
 49  t1_fighter          51490 non-null  int64
 50  t1_assassin         51490 non-null  int64
 51  t1_mage             51490 non-null  int64
 52  t1_support          51490 non-null  int64
 53  t1_marksman         51490 non-null  int64
 54  t2_tank             51490 non-null  int64
 55  t2_fighter          51490 non-null  int64
 56  t2_assassin         51490 non-null  int64
 57  t2_mage             51490 non-null  int64
 58  t2_support          51490 non-null  int64
 59  t2_marksman         51490 non-null  int64
dtypes: int64(60)
memory usage: 23.6 MB
```

Created new features by combining the original data with champions information dataframe (tags) - merge by ID

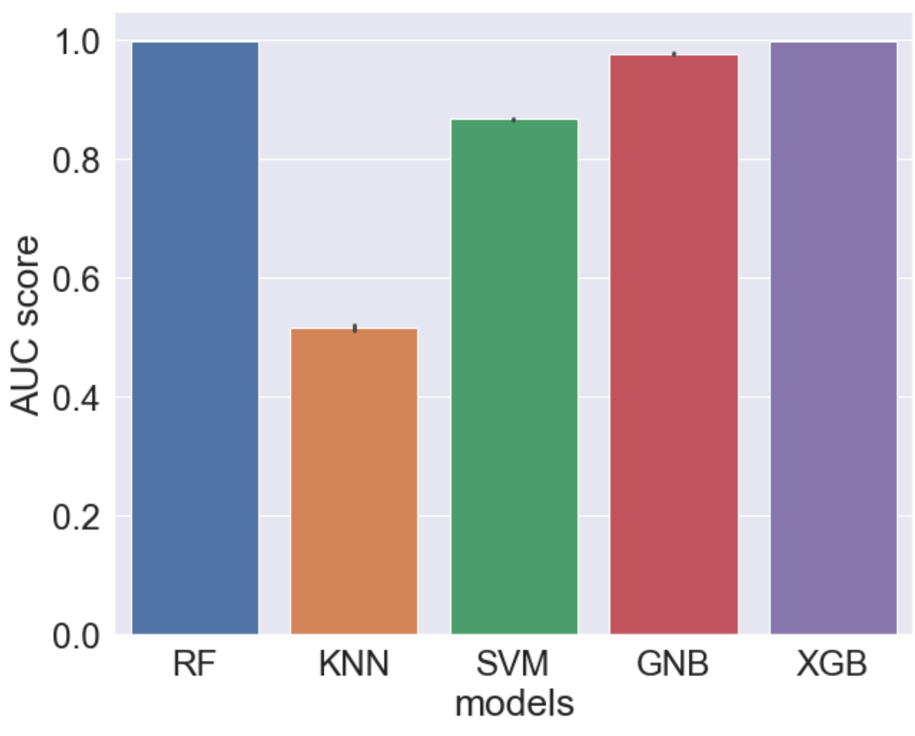
```python
# dropping redundant columns now that the new features have been created

lol_df.drop(['t1_tags', 't2_tags', 't1_champ1tags', 't1_champ2tags',\
             't1_champ3tags', 't1_champ4tags', 't1_champ5tags',\
             't2_champ1tags', 't2_champ2tags', 't2_champ3tags',\
             't2_champ4tags', 't2_champ5tags'], axis=1, inplace=True)

# dropping additional columns (bans) that may not be useful for ML model

lol_df.drop(['t1_ban1', 't1_ban2', 't1_ban3', 't1_ban4',\
             't1_ban5', 't2_ban1', 't2_ban2',\
             't2_ban3', 't2_ban4', 't2_ban5',\
             'gameId', 'creationTime', 'seasonId'], axis=1, inplace=True)
```

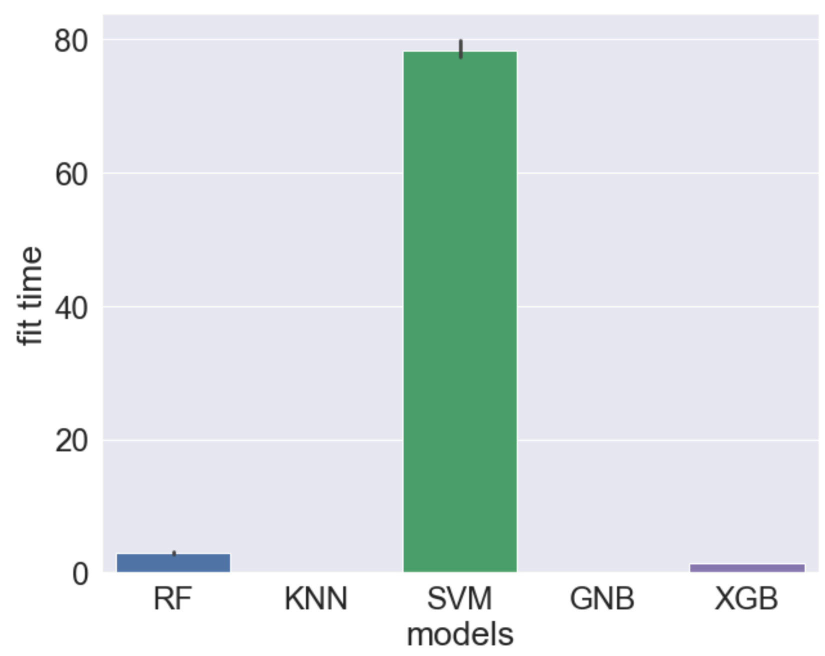
Dropped unnecessary columns

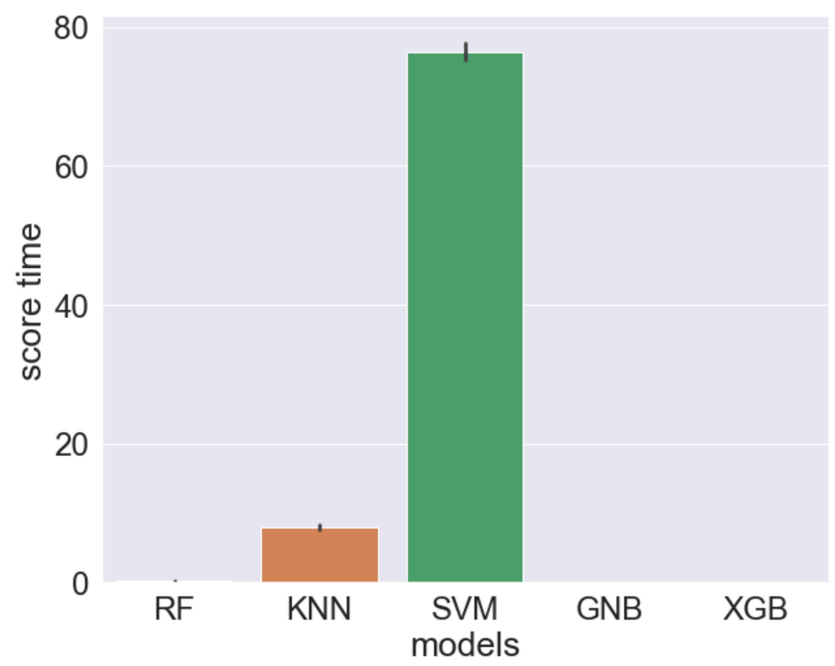
2. Prediction with various ML methods

From the different ML models,
Xgboost showed the best scores
in accuracy, precision, recall, and AUC.

However when it comes to fit and score time (duration), XGB was slower than KNN, SVM, and Gaussian Naive Bayes.







	fit_time	score_time	test_accuracy	test_precision_weighted	test_recall_weighted	test_f1_weighted	test_roc_auc	model
0	3.098676	0.156745	0.969535	0.969538	0.969535	0.969534	0.997483	RF
1	3.188755	0.158979	0.968079	0.968079	0.968079	0.968078	0.996986	RF
2	3.082980	0.155371	0.968318	0.968322	0.968318	0.968318	0.997245	RF
3	2.755910	0.145808	0.969896	0.969963	0.969896	0.969894	0.997281	RF
4	2.853027	0.146643	0.969289	0.969291	0.969289	0.969288	0.997106	RF
5	0.011879	7.443417	0.504673	0.504534	0.504673	0.504584	0.506498	KNN
6	0.009058	7.948629	0.514868	0.515096	0.514868	0.514908	0.517657	KNN
7	0.007834	8.609264	0.510682	0.510771	0.510682	0.510721	0.512282	KNN
8	0.008655	7.869237	0.519422	0.519359	0.519422	0.519251	0.525996	KNN
9	0.007799	8.021669	0.512017	0.511937	0.512017	0.511886	0.513218	KNN
10	81.408136	75.162202	0.774609	0.779443	0.774609	0.773123	0.865562	SVM
11	77.466676	78.443917	0.778614	0.784144	0.778614	0.777106	0.866595	SVM
12	77.507803	75.044874	0.766570	0.773918	0.766570	0.764190	0.865988	SVM
13	77.744422	75.576169	0.761835	0.776145	0.761835	0.758501	0.869118	SVM
14	77.192873	77.740065	0.765598	0.779628	0.765598	0.762365	0.865972	SVM
15	0.032172	0.026092	0.936036	0.936184	0.936036	0.936015	0.977248	GNB
16	0.039324	0.025562	0.936157	0.936207	0.936157	0.936148	0.975835	GNB
17	0.038388	0.025525	0.943190	0.943246	0.943190	0.943177	0.979070	GNB
18	0.045607	0.028353	0.936150	0.936419	0.936150	0.936135	0.974479	GNB
19	0.039907	0.025818	0.938820	0.939169	0.938820	0.938802	0.975311	GNB
20	1.345669	0.022050	0.971963	0.971998	0.971963	0.971965	0.997714	XGB
21	1.343268	0.023041	0.969778	0.969778	0.969778	0.969778	0.997153	XGB
22	1.372753	0.022921	0.970503	0.970505	0.970503	0.970503	0.997364	XGB
23	1.381449	0.022986	0.969531	0.969591	0.969531	0.969530	0.997235	XGB
24	1.497697	0.023089	0.971838	0.971839	0.971838	0.971838	0.997338	XGB

2. Prediction with various ML methods

	mean fit time	Mean score time	subsample	min child weight	max depth	gamma	mean test score	rank
3	31.693327	0.178384	1	5	5	5	0.997601	1
2	44.645198	0.182681	0.8	5	5	1	0.997599	2
1	48.195881	0.172682	0.6	1	5	1.5	0.997578	3
4	41.771196	0.127859	0.8	1	4	1	0.997563	4
0	30.222138	0.094446	1	5	3	5	0.997336	5

Hyperparameter tuning on Xgboost model

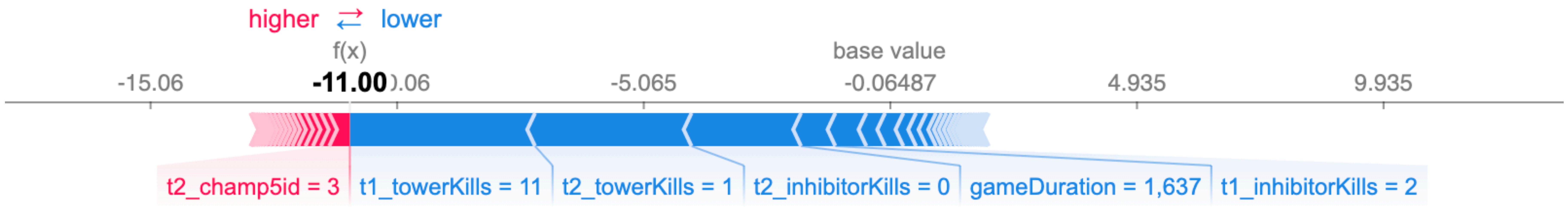
Best parameters:
subsample: 1.0, min_child_weight: 5,
max_depth: 5, gamma: 5, colsample_bytree: 0.6

Scoring metric : AUC

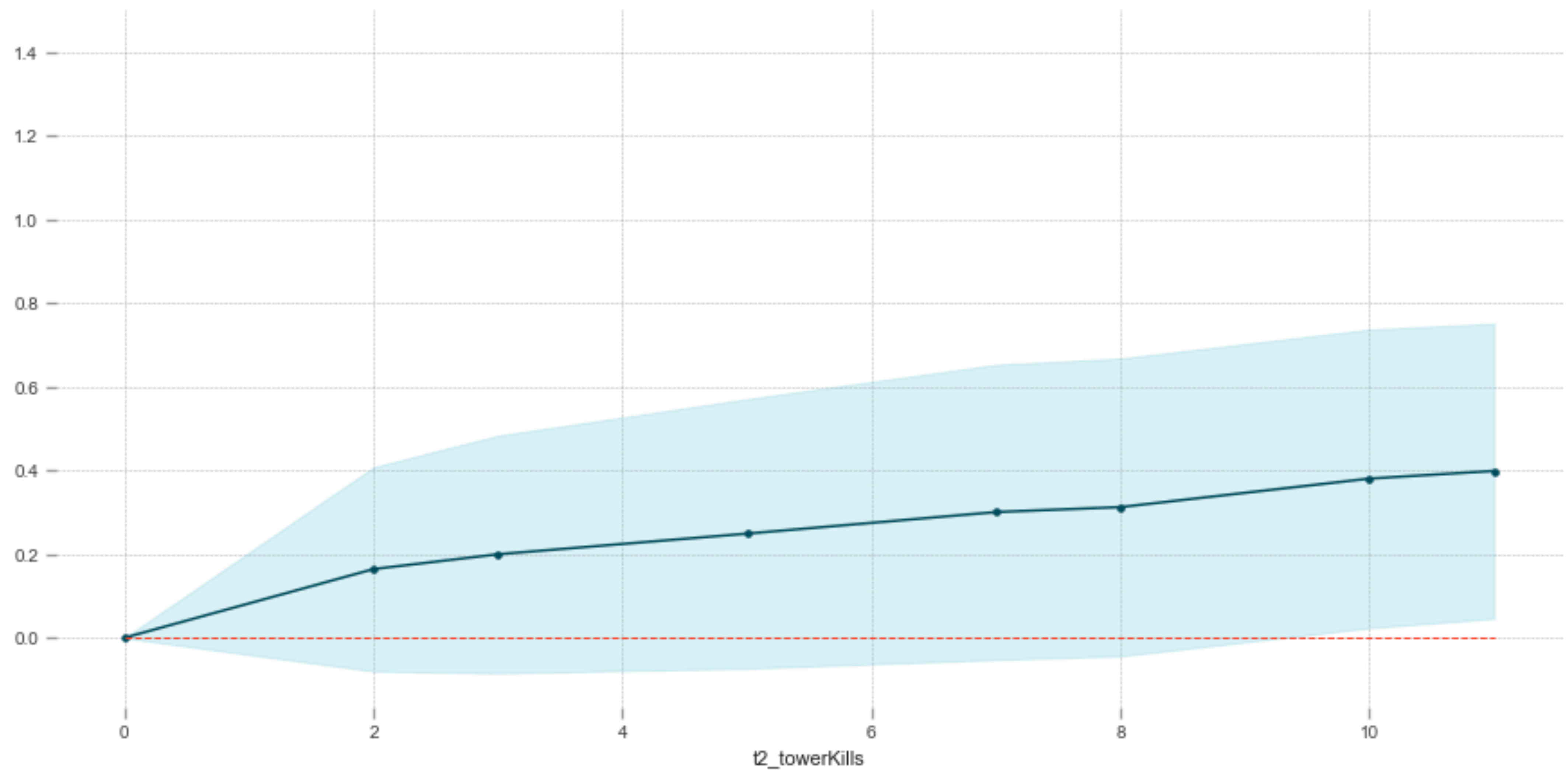
```
XGB with tuned parameters
              precision    recall  f1-score   support

      team 1       0.97      0.97      0.97      5147
      team 2       0.97      0.97      0.97      5151

    accuracy                           0.97     10298
   macro avg       0.97      0.97      0.97     10298
weighted avg       0.97      0.97      0.97     10298
```

3. Interpreting the model

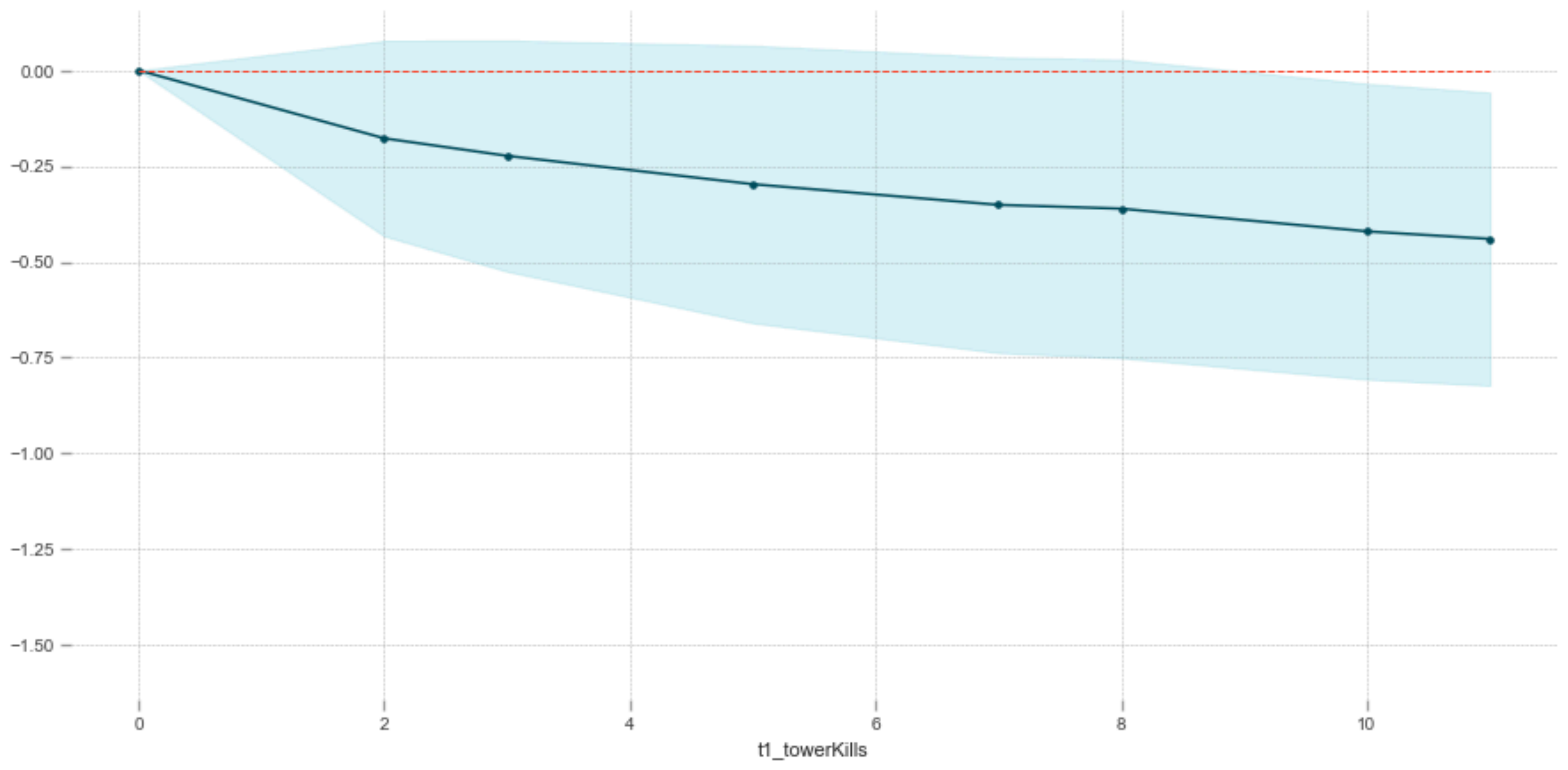
Shap values on XGB model :
Tower kills had the biggest impact on the model - data leakage?



PDP isolate plot for feature 'Team 2 Tower kills'



PDP isolate plot for feature 'Team 1 Tower kills'



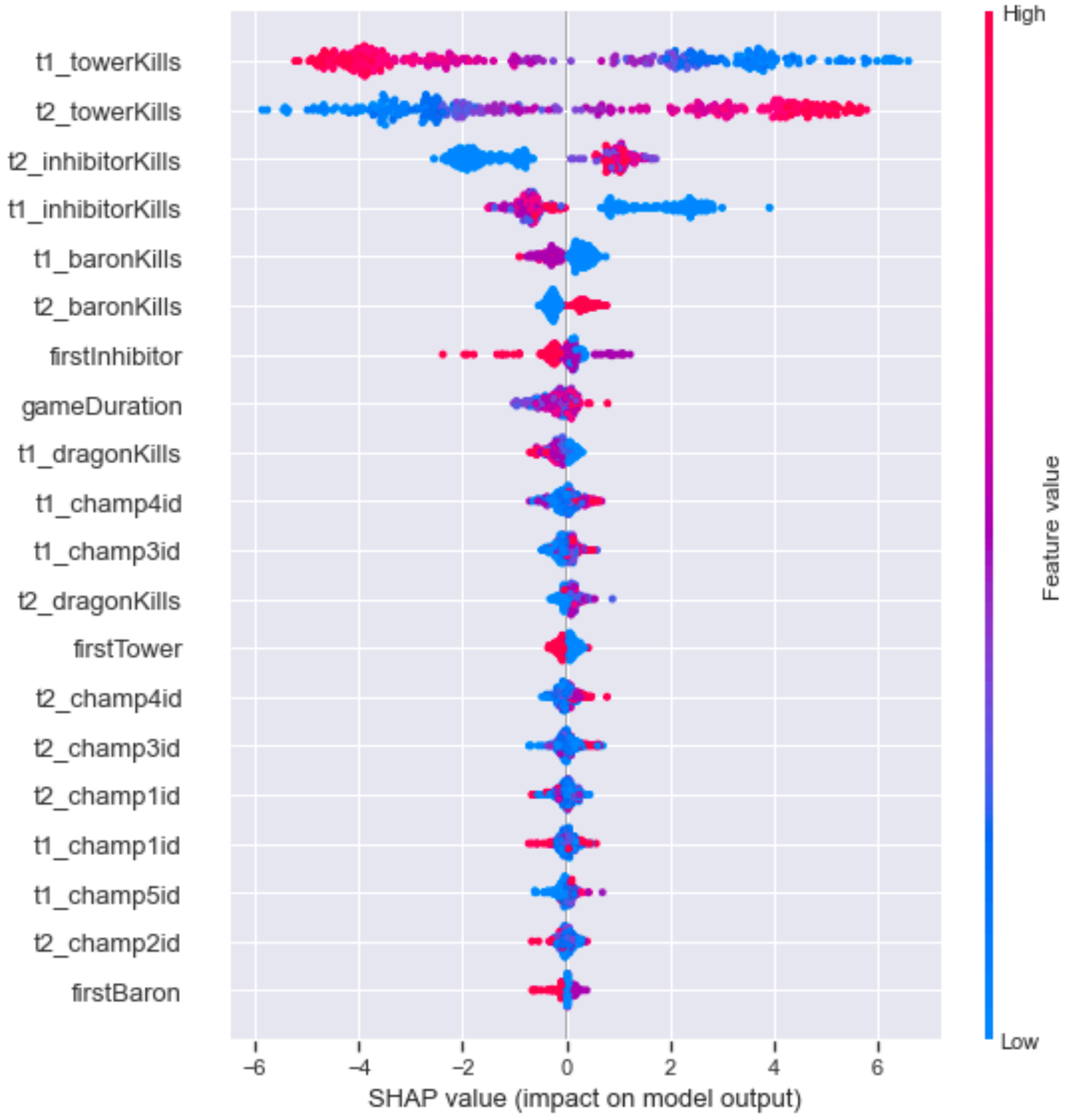
3. Interpreting the model

Dropped tower kills, inhibitor kills, and first inhibitor to prevent **data leakage**

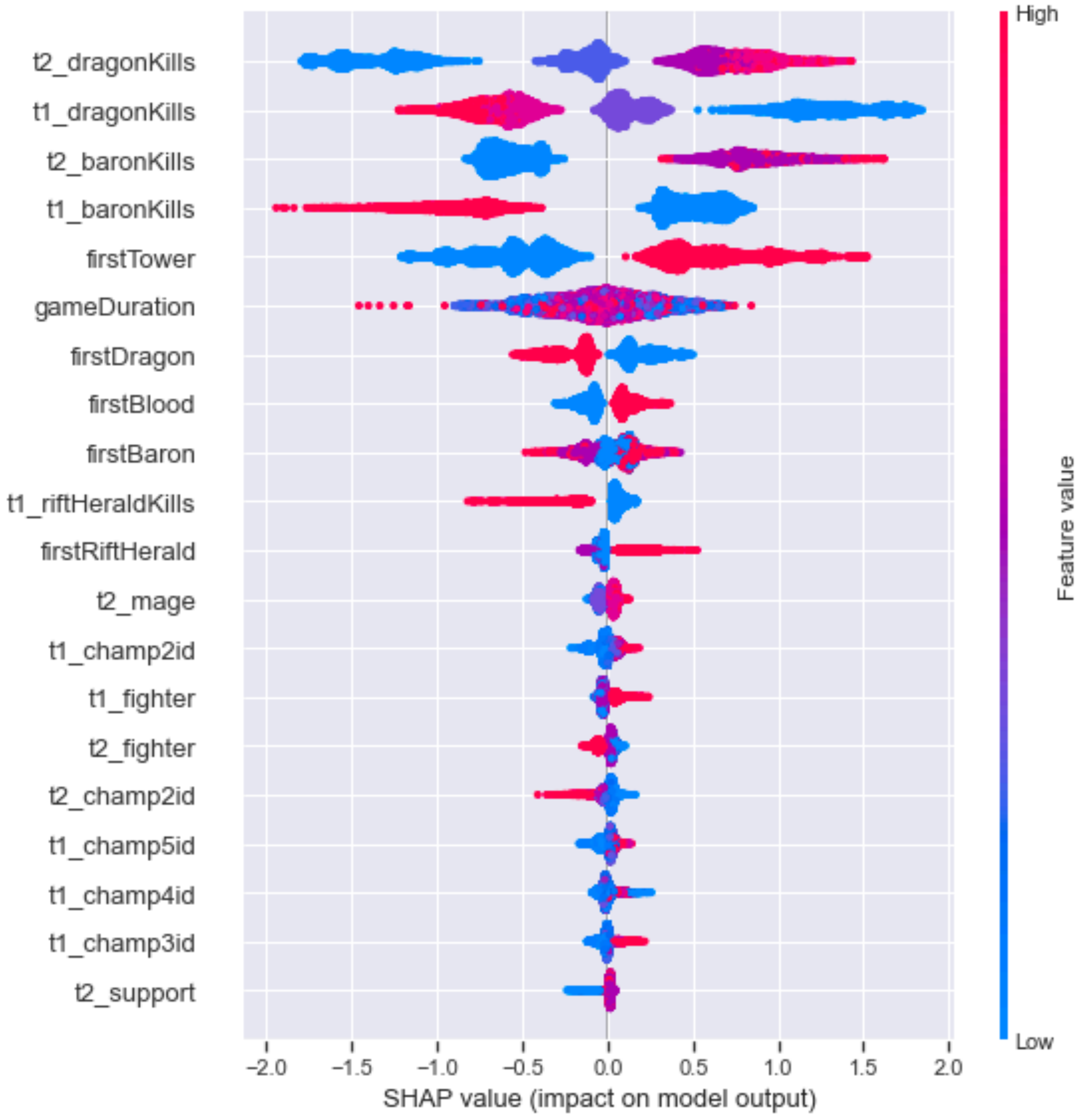
XGB with tuned parameters

	precision	recall	f1-score	support
team 1	0.86	0.86	0.86	5147
team 2	0.86	0.86	0.86	5151
accuracy			0.86	10298
macro avg	0.86	0.86	0.86	10298
weighted avg	0.86	0.86	0.86	10298

	precision	recall	f1-score	support
team 1	0.97	0.97	0.97	5147
team 2	0.97	0.97	0.97	5151
accuracy			0.97	10298
macro avg	0.97	0.97	0.97	10298
weighted avg	0.97	0.97	0.97	10298

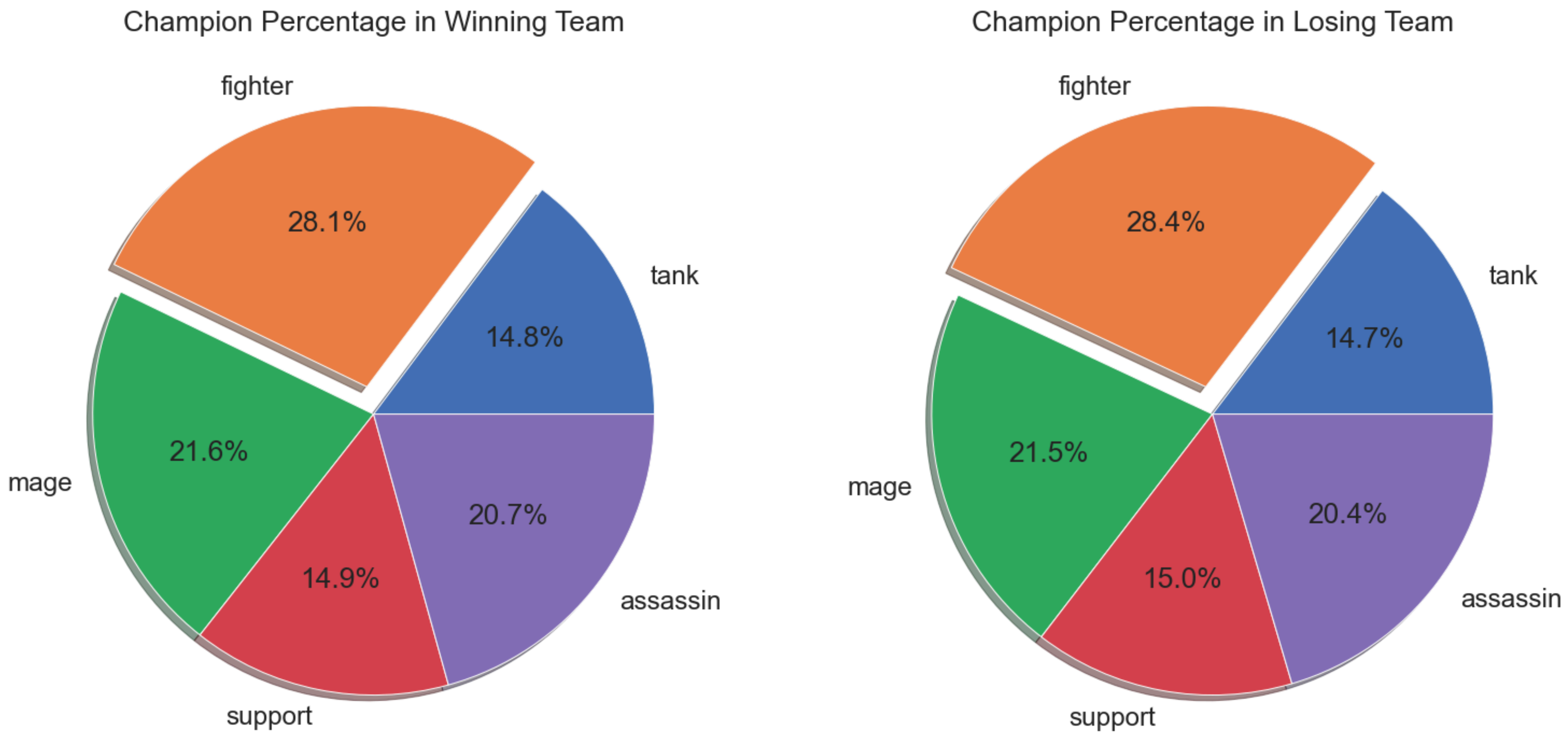


SHAP summary plot with data leakage



SHAP summary plot without data leakage

4. Analyzing champions of the winning team



Champion tags percentage in each team
(shows no significant difference)

```
t1_champs['combined'].value_counts()
```

```
[blitzcrank, kindred, kog'maw, renekton, tristana]    3
[dr. mundo, kayn, orianna, rakan, xayah]              3
[kha'zix, leona, lux, sivir, swain]                   3
[kayn, lissandra, miss fortune, rek'sai, taric]       3
[jarvan iv, jinx, master yi, thresh, yorick]          3
                                                     ..
[ahri, jayce, jhin, lee sin, zyra]                    1
[darius, jax, jinx, morgana, viktor]                  1
[ashe, kha'zix, rakan, thresh, vayne]                 1
[lulu, malphite, orianna, tryndamere, vayne]          1
[annie, janna, kindred, talon, tristana]              1
Name: combined, Length: 25803, dtype: int64
```

Frequent combinations from the Winning Team

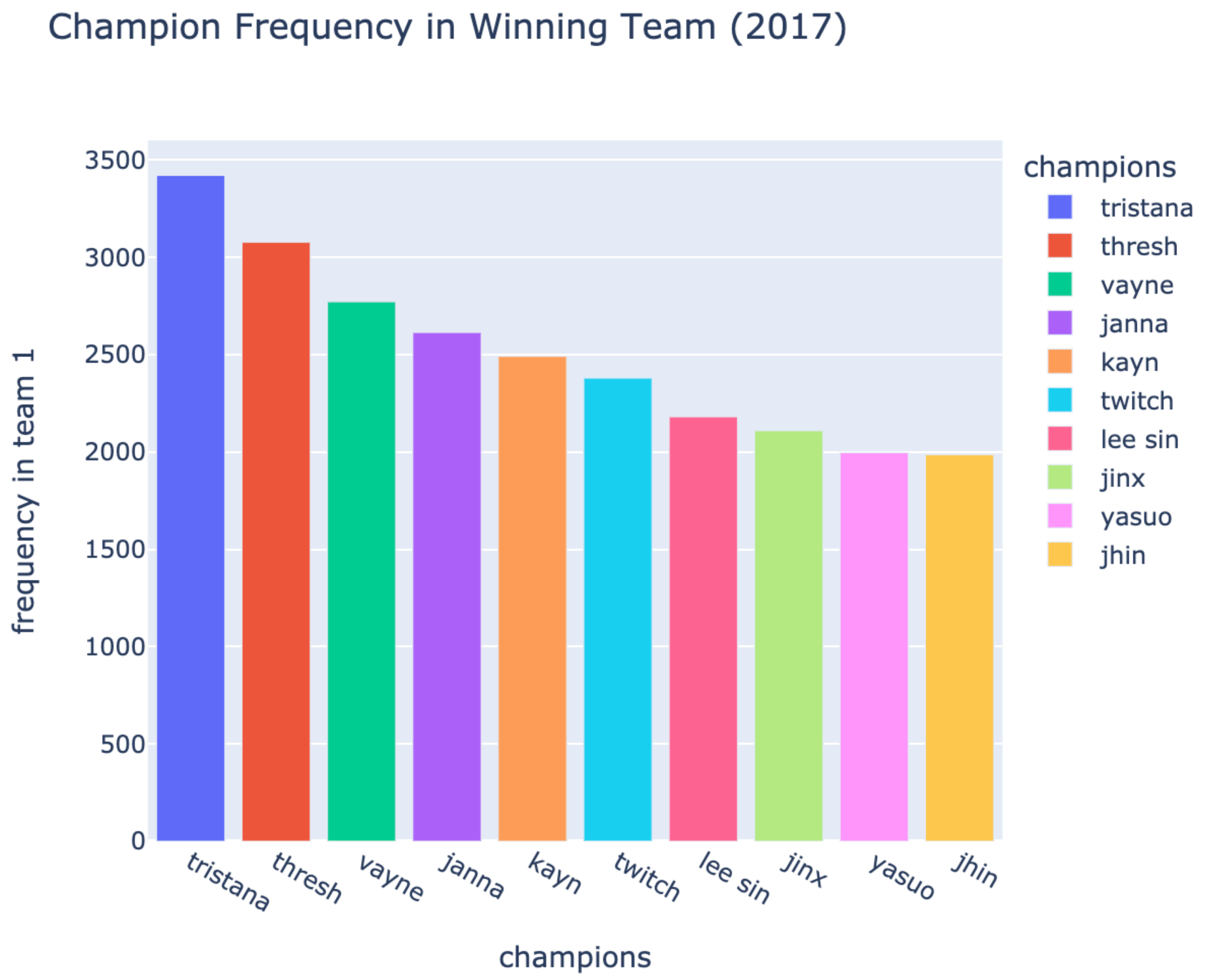
```
t2_champs['combined'].value_counts()
```

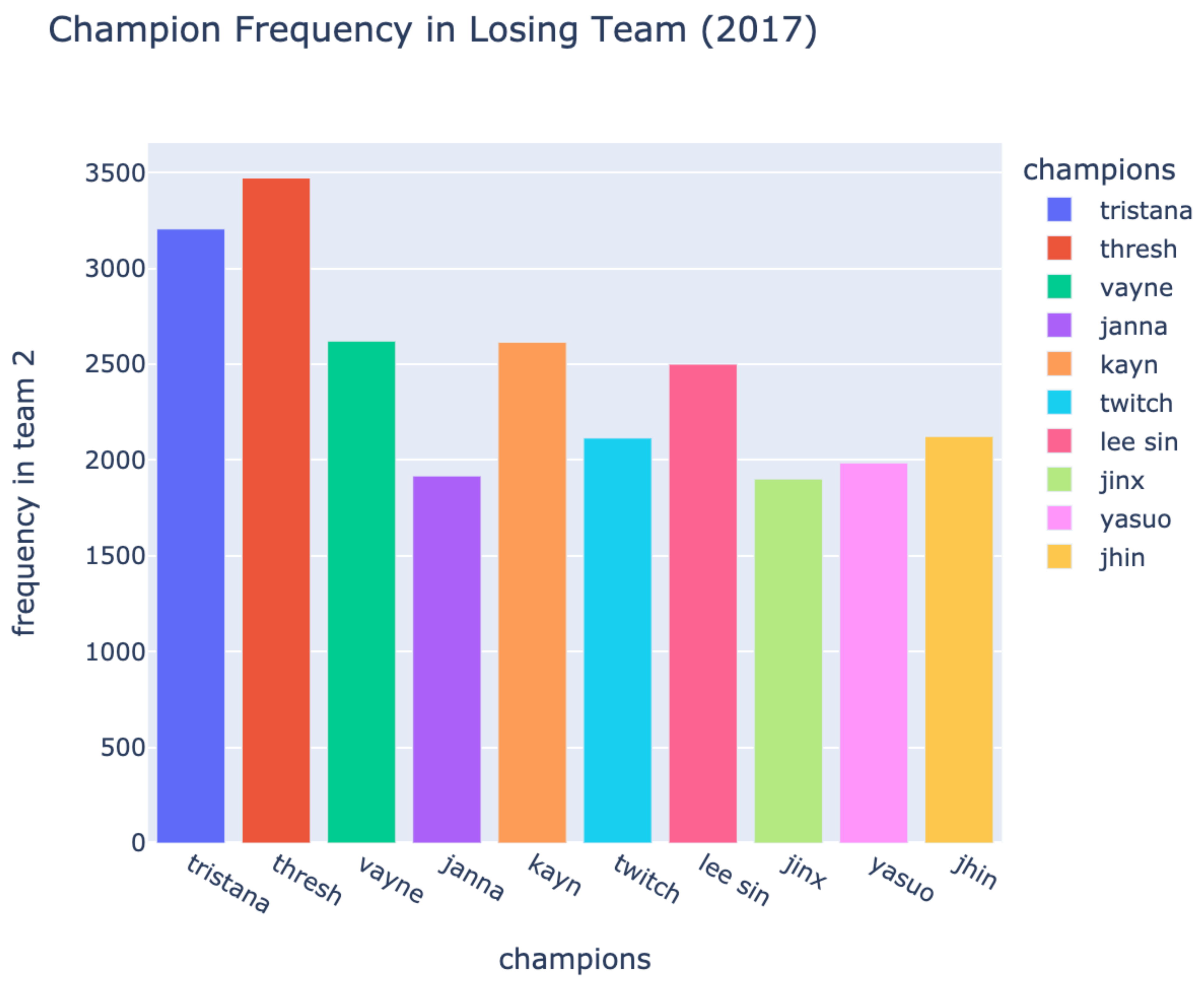
```
[lee sin, nami, swain, tristana, zed]              3
[jarvan iv, maokai, rakan, twisted fate, xayah]    3
[blitzcrank, jax, jhin, master yi, shaco]          3
[ezreal, lulu, orianna, thresh, twitch]            3
[diana, draven, janna, master yi, trundle]         3
                                                  ..
[ekko, rakan, rammus, tristana, yasuo]             1
[draven, evelynn, leblanc, maokai, zyra]           1
[blitzcrank, irelia, jarvan iv, syndra, varus]     1
[caitlyn, renekton, syndra, thresh, warwick]       1
[irelia, janna, jhin, malzahar, wukong]            1
Name: combined, Length: 25820, dtype: int64
```

Frequent combinations from the Losing Team

Champion Combinations frequency in each team
(no significant difference)

4. Analyzing champions of the winning team





Champion Frequency for the Top 10 Most played champion in each team (winning vs losing)

4. Conclusion

According to the previous data analysis,
We could observe following characteristics in League of Legends data (2017):

Tags	**No significant difference observed in different tag types such as Mage, Fighter, Tank Support, etc.**
Champions	• **Choosing Tristana** (0.52%) **over Thresh** (0.47%) **lead to higher winning possibility** • **Selecting Janna lead to victory for over 2600 teams** (800 teams more than the losing teams, 58% chance of winning) • **Selecting Lee Sin lead to defeat for 2500 teams** (Brought down the chance of winning to 0.46%)
Game Strategy	**Sequence of importance in Game Strategy** 1. Total dragon kills 2. Total baron kills 3. first tower 4. first dragon 5. first blood

4. Discussion

1. **Limitations in feature types :** Lack of data relevant to game strategy
   a. Champion position during the game (jungle, dragon, baron, mid, support)
   b. Champion's items, gold amount, Kill/Death/Support stat missing

2. **Lack of recent data :** 2017 (4 years ago)
   a. LoL is a fast-paced, competitive game that requires different strategies and champion performance.
   b. Viewing chronological data over different challenger tier players by region may be a more promising data analysis.