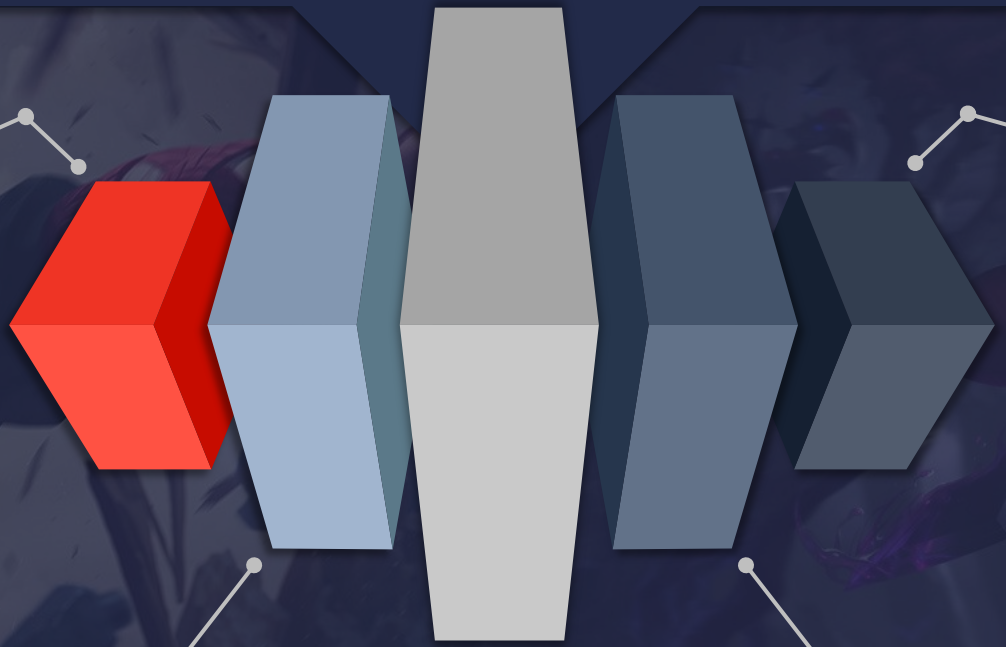
TEAM-BASED STRATEGY GAME



RED TEAM

Five Players per Team
Five Champions per Team
Objective: Destroy Blue team base

BLUE TEAM

Five Players per Team
Five Champions per Team
Objective: Destroy Red team base

MINION

Bots: Supportive role / pawns

CHAMPIONS

Over 140 champions
Different skills, strengths and weaknesses

TOWERS

Six towers per team (Forts)

Champion Roles

FARM. KITE. CARRY

AD CARRY

SUPPORT

PROTECT. HARASS. DOMINATE

SMITE. GANK. TAX

JUNGLE

TOP

PUSH. CARRY. ROAM

THE WORLD IS A DESERT.
I AM THE OASIS

MID

CREATE A SUPERSQUAD

ANALYSIS WORKFLOW

STEP1

Identify Data Source

STEP2

Clean and Explore Data

STEP3

Perform Initial Descriptive Data Analysis

STEP4

Conduct Correlation Analysis

STEP5

Create Algorithm to Calculate Viability Score

Data Sources

Analysis Workflow

Champions

Season: ALL S4 S5 S6 S7 S8 S9 S10

Split: ALL Pre-Season Spring Summer

Tournament: -- ALL --

Game version: -- ALL --

Role: ALL TOP JUNGLE MID BOT SUPPORT

Leagues: TOP TIER2 NAT

Copy table to clipboard

CHAMPION	PICKS	BANS	PRESENCE	AVG BT	WINS	LOSSES	WINRATE
Aphelios	259	170	86%	4.7	140	119	54%
Varus	38	326	73%	2.5	19	19	50%
Kalista	103	260	73%	3.4	51	52	50%
Syndra	117	242	72%	4.2	60	57	51%
Ezreal	219	130	70%	4.8	110	109	50%
Twisted Fate	119	210	66%	4.6	71	48	60%

User Champion Picks 2018-2020

Games of Legends eSPORTS

Data Sources

Analysis Workflow

Champion stats for majority of analysis

CHAMPION.GG

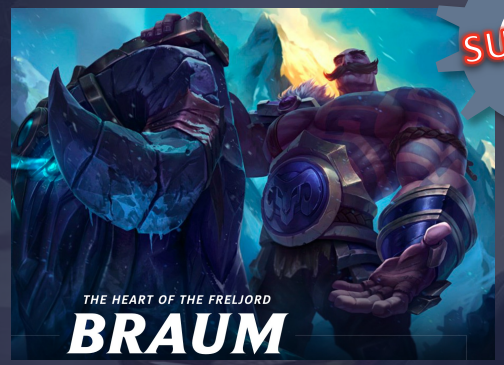
Current Patch Statistics

Filter By Name | Sort Role: Show All

Rank	Champion	Role	Win Percent	Play Percent	Ban Rate	Playerbase Avg. Games	Kills	Deaths	Assists	Largest Killing Spree	Damage Dealt	Damage Taken	Total Healing	Minions Killed	Enemy Jungle CS	Team Jungle CS	Gold Earned	Role Position	Position Change▼
1	Swain	Middle	54.04%	0.80%	0.27%	9.81	6.87	5.64	9.15	6	23692	29502	10086	152.0	1.29	3.79	11090	7	0
2	Skarner	Top	45.27%	0.21%	0.03%	12.75	4.24	5.31	7.56	4	13155	22879	2086	138.5	1.64	2.60	9951	45	0
3	Senna	ADC	52.09%	2.04%	1.32%	NaN	5.89	5.18	11.29	7	19715	17444	10538	156.3	0.96	4.14	11301	16	0
4	Riven	Support	49.44%	0.54%	0.15%	54.61	3.22	3.44	1.69	4	6865	10187	866	101.4	1.10	1.05	6440	64	0
5	Lillia	Jungle	42.73%	7.09%	1.94%	NaN	5.91	5.82	7.77	9	17111	28494	9469	42.7	7.07	84.67	10596	17	0
6	Lillia	Top	45.09%	1.48%	1.94%	NaN	5.13	6.14	6.65	7	19620	26316	4748	159.0	1.78	3.36	10667	36	0
7	Karthus	ADC	55.00%	0.48%	1.57%	34.85	9.01	7.73	9.93	8	30765	21845	5654	177.7	3.04	11.23	13017	1	0
8	Veigar	Support	50.68%	0.51%	0.11%	7.59	3.94	5.24	6.28	6	10897	12055	1067	68.1	0.58	1.12	7744	45	↑22
9	Illaoi	Top	50.14%	1.14%	0.37%	5.35	5.26	6.28	4.67	6	23148	30556	8819	191.2	1.31	1.30	11686	8	↑20
10	Qiyana	Support	54.94%	0.26%	0.11%	NaN	4.77	3.40	2.86	7	8727	9615	1423	93.7	0.70	0.69	6716	32	↑19

Data Exploration and Cleaning

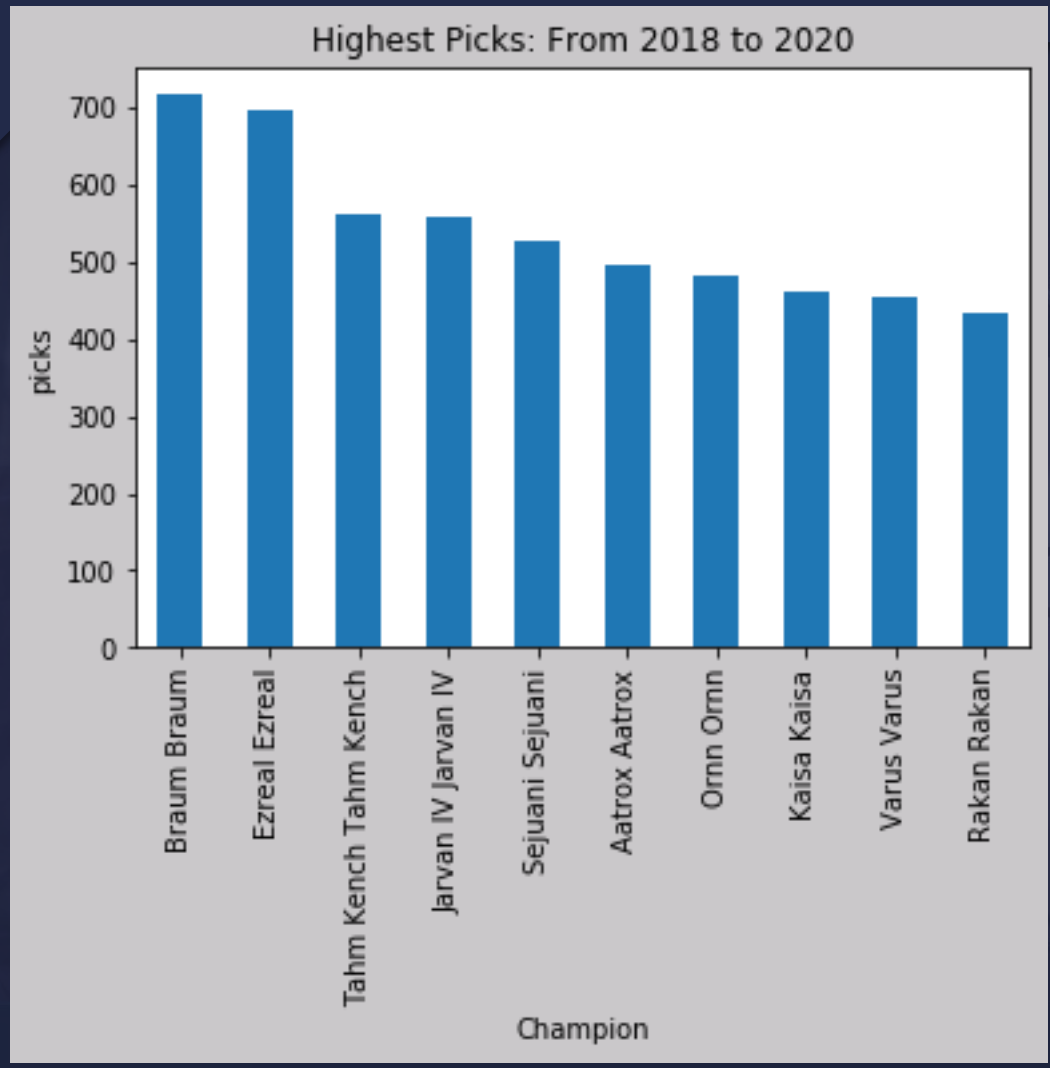
Analysis Workflow



SUPPORT

ADC / TOP

```
reduced_champions_df = champions_df.groupby(["Champion"]).mean()

#reduced_champions_df.head()
new = reduced_champions_df.sort_values(["picks"],ascending=False)["picks"]
healing_graph = new.head(10).plot(kind="bar")
healing_graph.set_ylabel("picks")
plt.title("Highest Picks: From 2018 to 2020")
plt.show()
```



Data Exploration and Cleaning

Analysis Workflow

```
#Columns in the dataset
stack_df.columns

#Search for null values, number of non-null values, data types
stack_df.info()

#Identify Null Values
df_null = stack_df[stack_df.isnull().any(axis=1)]

#Drop weird rows with no Champion listed & check non-null values
df2 = stack_df.dropna(subset = ['Champion'])
df2.info()

## === Notice that playerbase avg games still has less data (898) === ##
```

	Rank	Champion	Role	Win Percent	Play Percent	Ban Rate	Playerbase Avg. Games	Kills	De
0	1	Aatrox	Top	46.90%	2.06%	0.42%	2.58	5.49	
1	2	Ahri	Middle	51.84%	3.33%	0.30%	2.91	6.82	
2	3	Akali	Middle	43.79%	3.49%	0.77%	2.98	8.45	
3	4	Akali	Top	46.67%	1.27%	0.77%	3.98	7.21	
4	5	Alistar	Support	51.76%	2.43%	0.07%	2.41	2.02	
...	...	...	...	...	...	...	...	...	...
183	184	Zilean	Support	51.21%	1.74%	0.12%	3.05	2.57	
184	185	Zoe	Middle	46.81%	1.85%	0.31%	NaN	6.62	
185	186	Zyra	Support	52.83%	2.61%	0.27%	4.01	3.75	
186	Champion.gg isn't endorsed by Riot Games and d...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
187	Blitz App • FortniteMaster • StatsRealmRoyale ...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

9 rows × 21 columns

YIKES!!

Data Exploration and Cleaning

Analysis Workflow

```
#Columns in the dataset
stack_df.columns

#Search for null values, number of non-null values, data types
stack_df.info()

#Identify Null Values
df_null = stack_df[stack_df.isnull().any(axis=1)]

#Drop weird rows with no Champion listed & check non-null values
df2 = stack_df.dropna(subset = ['Champion'])
df2.info()

## === Notice that playerbase avg games still has less data (898) === ##
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 989 entries, 0 to 187
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Rank                    989 non-null    object
 1   Champion                979 non-null    object
 3   Win Percent             979 non-null    object
 4   Play Percent            979 non-null    object
 5   Ban Rate                979 non-null    object
 6   Playerbase Avg. Games   898 non-null    float64
 7   Kills                   979 non-null    float64
 8   Deaths                  979 non-null    float64
 9   Assists                 979 non-null    float64
 10  Largest Killing Spree   979 non-null    float64
 11  Damage Dealt            979 non-null    float64
 12  Damage Taken            979 non-null    float64
 13  Total Healing           979 non-null    float64
 14  Minions Killed          979 non-null    float64
 15  Enemy Jungle CS         979 non-null    float64
 16  Team Jungle CS          979 non-null    float64
 17  Gold Earned             979 non-null    float64
 18  Role Position           979 non-null    float64
 19  Position Change         979 non-null    float64
 20  League                  989 non-null    object
dtypes: float64(14), object(7)
```

Data Exploration and Cleaning

Analysis Workflow

```
#Percentage values showing up as OBJECTS,
#problematic, need to remove the % for calculations!
champ_df = df2.groupby(["Champion"]).mean()
champ_df.head()

## === Notice there are NO stats for these metrics ==== ##
```

Champion	Playerbase Avg. Games	Kills	Deaths	Assists	Largest Killing Spree	Damage Dealt	
Aatrox	8.174286	4.644286	4.957143	4.828571	6.000000	15734.857143	2
Ahri	8.608333	6.106667	5.121667	7.061667	6.666667	18547.666667	1
Akali	12.318333	6.916667	5.678333	4.551667	7.500000	17378.416667	2
Alistar	3.300000	1.910000	5.650000	12.508000	3.800000	7564.200000	2
Amumu	2.844000	4.554000	5.704000	9.666000	6.800000	13318.600000	2

```
#Create a function to convert percent to float
def convert_percent(val):
    """
    Convert the percentage string to an actual floating point percent
    - Remove %
    - Divide by 100 to make decimal
    """
    new_val = val.replace('%', '')
    return float(new_val) / 100

# Created a new column that applies the function to convert percentage to float
df2['win_percent'] = df2["Win Percent"].apply(convert_percent)
df2['ban_rate'] = df2["Ban Rate"].apply(convert_percent)
df2['play_percent'] = df2["Play Percent"].apply(convert_percent)

# Drop extra columns
df2.drop(columns=['Win Percent', 'Play Percent', 'Ban Rate'])
```

Data Exploration and Cleaning

Analysis Workflow

```python
#Create a function to convert percent to float
def convert_percent(val):
    """
    Convert the percentage string to an actual floating point percent
    - Remove %
    - Divide by 100 to make decimal
    """
    new_val = val.replace('%', '')
    return float(new_val) / 100

# Created a new column that applies the function to convert percentage to float
df2['win_percent'] = df2["Win Percent"].apply(convert_percent)
df2['ban_rate'] = df2["Ban Rate"].apply(convert_percent)
df2['play_percent'] = df2["Play Percent"].apply(convert_percent)

# Drop extra columns
df2.drop(columns=['Win Percent', 'Play Percent', 'Ban Rate'])
```

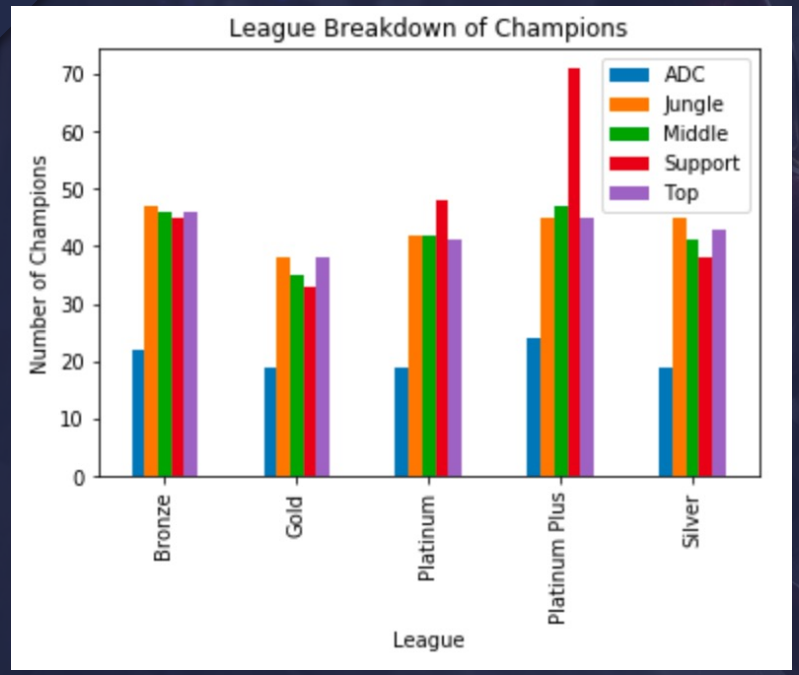
	Rank	Champion	Role	League	win_percent	ban_rate	play_percent	Kills	Deaths	Assists	Largest Killing Spree
0	1	Aatrox	Top	Bronze	0.4690	0.0042	0.0206	5.49	5.90	5.96	7.0
1	2	Ahri	Middle	Bronze	0.5184	0.0030	0.0333	6.82	5.69	7.73	7.0
2	3	Akali	Middle	Bronze	0.4379	0.0077	0.0349	8.45	6.61	5.48	10.0
3	4	Akali	Top	Bronze	0.4667	0.0077	0.0127	7.21	6.06	4.82	7.0
4	5	Alistar	Support	Bronze	0.5176	0.0007	0.0243	2.02	5.84	12.80	4.0

Data Exploration and Cleaning

Analysis Workflow

	Feature	Unique values
0	Role	5
1	League	5
2	Largest Killing Spree	10
3	Position Change	21
4	Role Position	71
5	Champion	148
6	ban_rate	215
7	Rank	232
8	Deaths	287
9	play_percent	470
10	Kills	515
11	Assists	547
12	win_percent	552
13	Minions Killed	645
14	Gold Earned	869
15	Total Healing	922
16	Damage Dealt	947
17	Damage Taken	950

```
#Explore Unique Values in the Dataset
ind_list = []
unique_count = []
for col in df2.columns:
    ind_list.append(col)
    unique_count.append(len(df2[col].unique()))
final_series = pd.DataFrame({'Feature':ind_list,'Unique values':unique_count})
                        .sort_values(by='Unique values',ascending=True)
                        .reset_index(drop=True).style.bar()
final_series
```




```
#Create a graph that shows the number of Champions per Role with League as the X axis
role_chart = df2.groupby(["League", "Role"]).count()["Rank"].unstack().plot(kind = "bar")
role_chart.set_ylabel("Number of Champions")
plt.title("League Breakdown of Champions")
plt.legend(loc="best")
plt.show()
```

Descriptive Data Analysis

Data Workflow

Support

- High total healing
- High damage taken
- High minion kills
- Low death
- High assist

ADC

- High gold earned
- High damage taken
- High damage dealt

Middle

- High kills
- Low deaths
- High assist
- High minions kills

Jungle

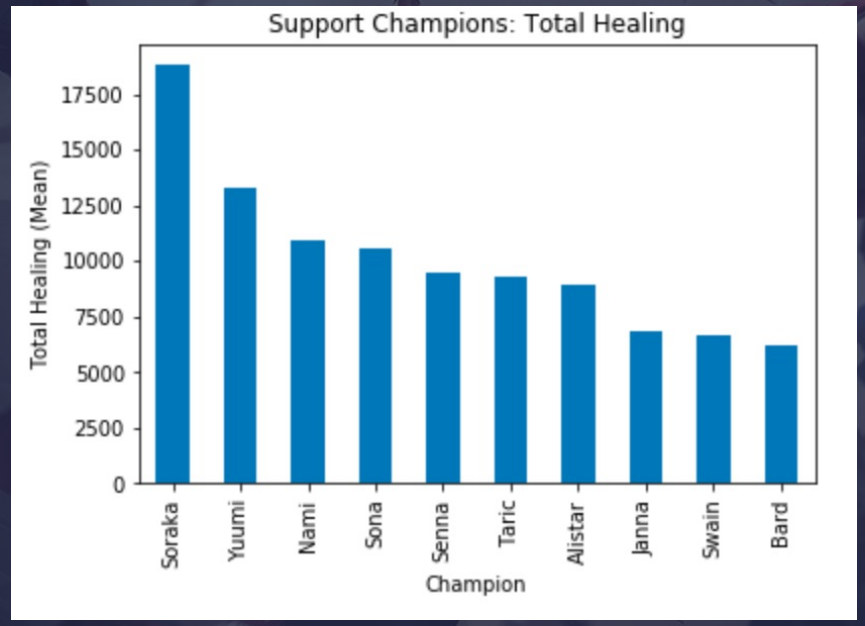
- High minion kills
- High enemy jungle
- High team jungle cs
- High kills
- High assist
- High position change

Top

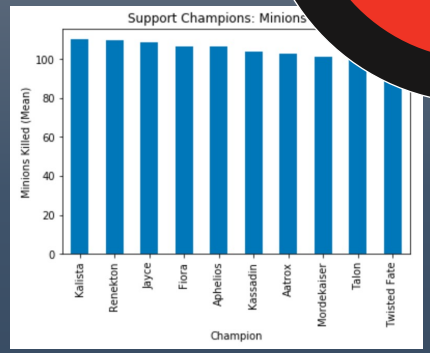
- High kills
- High kills spree
- High minions kills
- Low deaths

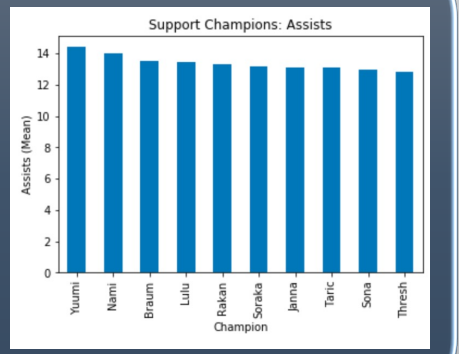
Descriptive Data Analysis

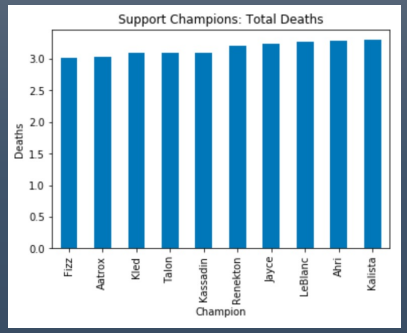
Data Workflow




```
#Bar chart for healing
healing = champ_support.sort_values(["Total Healing"],ascending=False)["Total Healing"]
healing_graph = healing.head(10).plot(kind="bar")
healing_graph.set_ylabel("Total Healing (Mean)")
plt.title("Support Champions: Total Healing")
plt.show()
```







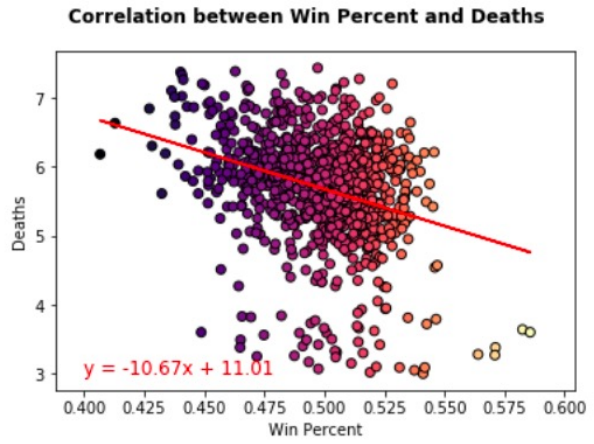
No ONE Champion stands out

Correlation Data Analysis

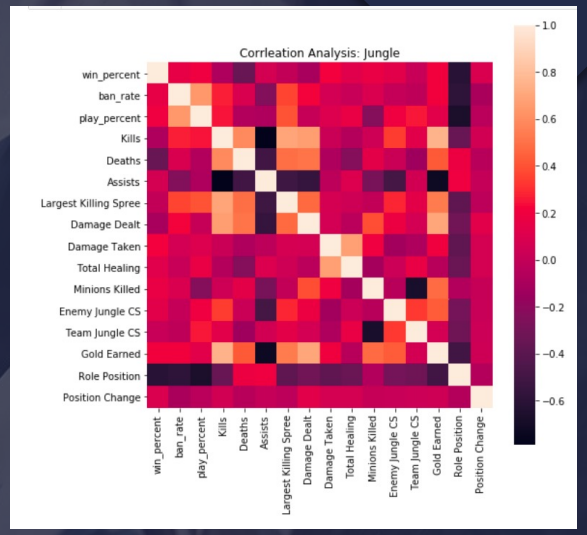
Analysis Workflow

Pearson's

```
The r-squared is: 0.09804673350597995
```

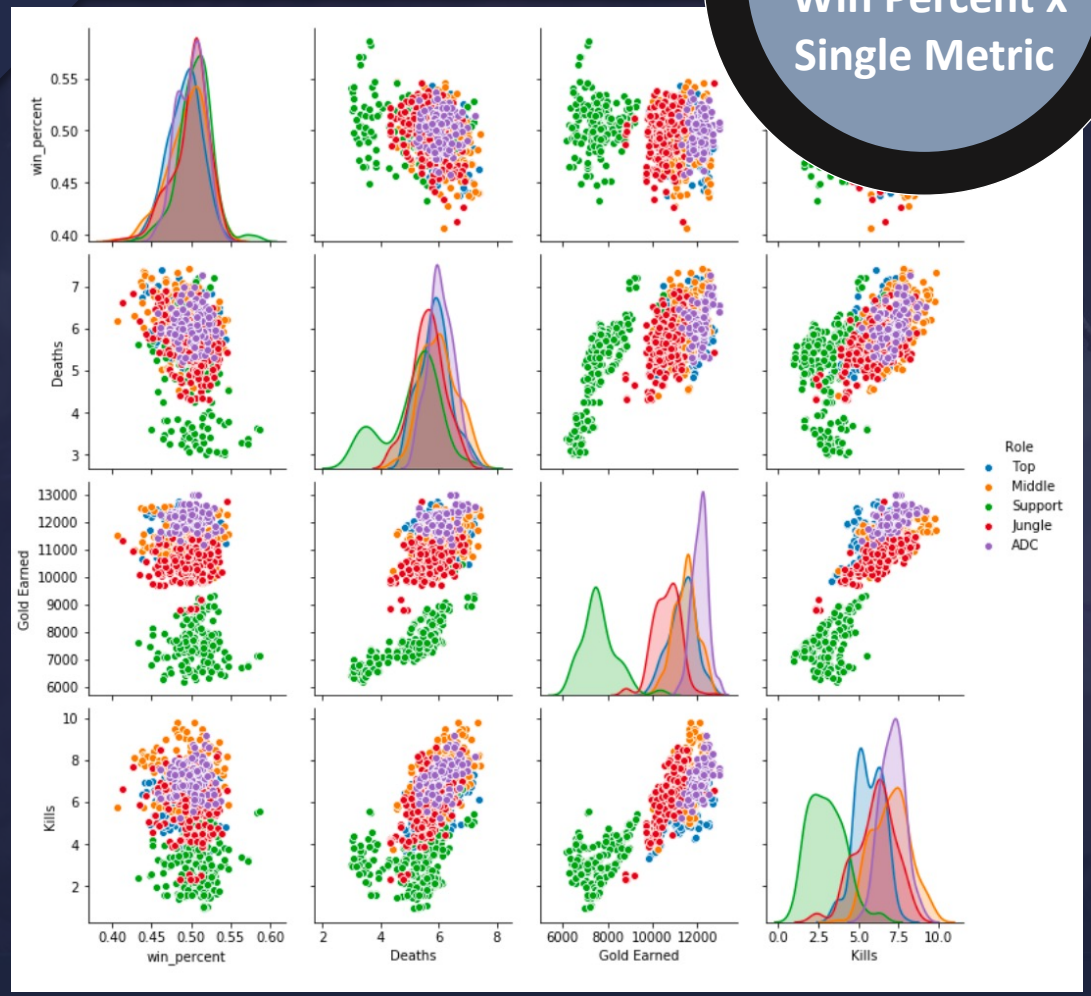



```
x_win = df2["win_percent"]
y_deaths = df2["Deaths"]

#slope calculations
(slope, intercept, rvalue, pvalue, stderr) = linregress(x_win, y_deaths)
regress_values = x_win * slope + intercept
line_eq = "y = " + str(round(slope,2)) + "x + " + str(round(intercept,2))
plt.ylabel('Deaths')
plt.xlabel('Win Percent')


t=x_win
plt.scatter(x_win, y_deaths, marker="o", edgecolor = "black", c=t, cmap='magma')
plt.suptitle("Correlation between Win Percent and Deaths", fontsize=12, fontweight = "bold")
plt.plot(x_win, regress_values, "r-")
plt.annotate(line_eq,(0.4,3),fontsize=12, color = "red")

print(f"The r-squared is: {rvalue**2}")
plt.show()
```




```
jungle = df2.loc[df2["Role"] == "Jungle"]
plt.figure(figsize=(8,8))
sns.heatmap(jungle.corr(),annot=False,square=True)
plt.title("Corrleation Analysis: Jungle")
plt.show()
```

No Strong Correlation Win Percent x Single Metric




```
#Subset of Data for Presentation
sub2_df2 = df2[["win_percent", "Role", "Deaths", "Gold Earned", "Kills"]]
sns.pairplot(sub2_df2, hue="Role")
```

Viability Score

Analysis Workflow

```
# Running calc to find the most viable Support
top_score = 0
top_name = ''
current_score = 0
for index, row in support.iterrows():
    current_score = (row['Minions Killed'] - row['Deaths']
                     + row['Total Healing'] + row['Damage Taken']
                     + row['Assists'])

    if current_score > top_score :
        top_score = current_score
        top_name = row['Champion']
print(f'{top_name} is the most viable Support with a score of {int(top_score)}')
```

```
Soraka is the most viable Support with a score of 31853
```

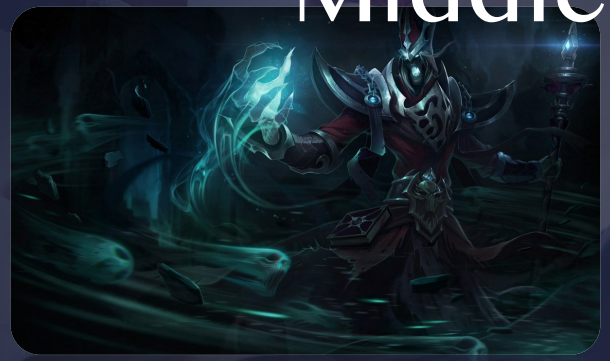
```
# Running calc to find most viable Top
top_score = 0
top_name = ''
current_score = 0
for index, row in top.iterrows():
    current_score = (row['Minions Killed'] - row['Deaths']
                      + row['Kills'] + row['Largest Killing Spree'])

    if current_score > top_score :
        top_score = current_score
        top_name = row['Champion']
print(f'{top_name} is the most viable Top with a score of {int(top_score)}')
```

```
Irelia is the most viable Top with a score of 208
```

This equation was designed to take the most important metrics and use them to create an overall viability score. Using this score we can find the most effective champion in their respective role.

Super Squad

Middle



Karthus

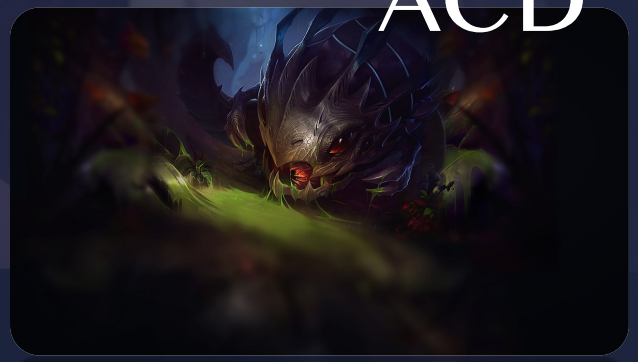
Top

Irelia

Jungler

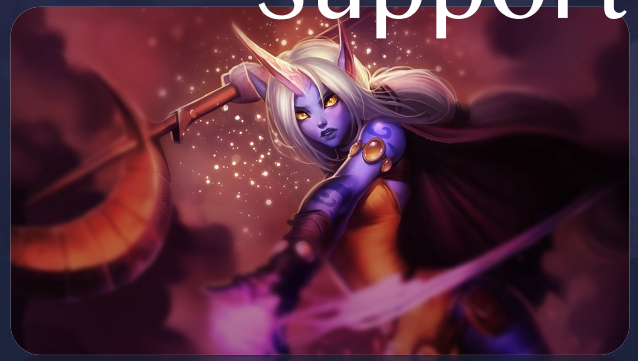
Signed

ACD



Kog'Maw

Support



Soraka

Analysis Summary

Conclusions and Post-Mortem

Dataset

Utilized Kaggle datasets from second-hand APIs
Required mild cleaning and merging of datasets

Analysis

Top 10 analysis did not identify obvious champions
Correlation analysis provided insight into roles but no strong correlation with win percent

Viability Score

Rather than look at single metric, we combined our knowledge of the game with key metrics to create a viability score

Improving Analysis

1. Utilize Riot API for real-time data
2. More statistics on game play rather than character attributes