FINAL PROJECT

BYRON WASHINGTON

BACKGROUND

- I’ve been playing League of Legends for a while now
- I play with bad players (my friends)
- They ask: “Who is good”
- I will answer that

OBJECTIVES

Top 2% of Players

We want to find a handful of champions that are in the best roles and are easy to pick up.

To do this we're looking for a few things:

- Win Rate
- Ban Rate
- Champion Difficulty
- Average KDA (Kills/Deaths/Assists)
- Role Win Rate
- Role Difficulty
- Resource Types

Cleaning the Data

- Importing packages and dataset

```
import matplotlib.pyplot as plt # plotting
import seaborn as sea # plotting
import pandas as pd # data processing

```

- Dropping columns that aren't needed

```
# initializing
path = '../lol_champions.csv'
df = pd.read_csv(path, delimiter=';', encoding_errors='ignore')

df.drop(['blurb', 'title', 'ulti_description', 'spell1_description', 'spell2_description', 'spell3_description',
         'ulti_name', 'spell1_name', 'spell2_name', 'spell3_name','ulti_cost', 'spell1_cost', 'spell2_cost', 'spell3_cost',
         'ulti_cooldown', 'spell1_cooldown', 'spell2_cooldown', 'spell3_cooldown', 'pentas/match', 'key', 'releasedate',
         'attack', 'defense', 'magic', 'popularity_spot', 'id', 'tags', 'rangetype', 'hp', 'hpperlevel', 'mp', 'mpperlevel',
         'movespeed', 'armor', 'armorperlevel', 'spellblock', 'spellblockperlevel', 'attackrange', 'hpregen', 'hpregenperlevel',
         'mpregen', 'mpregenperlevel', 'attackdamage', 'attackdamageperlevel', 'attackspeed', 'attackspeedperlevel', 'popularity'], axis='columns', inplace=True)
df.head()
```

✓ 0.0s

	name	winrate	banrate	averageKDA	difficulty	role	partype
0	Aatrox	49.5%	17.3%	6.1/6.1/5.3	4.0	Top	Blood Well
1	Ahri	51.0%	4.4%	6.2/5.3/7.6	5.0	Middle	Mana
2	Akali	49.3%	9.3%	8.4/5.8/4.8	7.0	Top,Middle	Energy
3	Akshan	48.6%	1.9%	8.5/6.5/5.5	7.0	Middle	Mana
4	Alistar	49.0%	2.0%	1.9/6.4/13.8	7.0	Support	Mana

Looking at the table, most of the formatted data doesn't look right so I had to clean a few things:

- Fixing the resource types
- Changing "averageKDA" to be one ratio
- Converting 'role' into an array of roles
- Parsing "winrate" and "banrate"

```
# Fixing resource types for everyone
df.loc[df['name'].isin(['Briar', 'Dr. Mundo', 'Vladimir', 'Zac']), 'partype'] = 'Health'
df.loc[~df['partype'].isin(['Mana', 'Energy', 'Health']), 'partype'] = 'None'

# Changing Kills/Deaths/Assists into one KDA ratio
def findKDA(row):
    parsedNums = row.split('/')
    parsedNums = [float(x) for x in parsedNums]
    return (parsedNums[0]+ parsedNums[2]) / parsedNums[1]
df['averageKDA'] = df['averageKDA'].apply(findKDA)

# Converting each role into an array of roles
df['role'] = df['role'].str.split(',')

# Parsing Win Rate and Ban Rate
df['winrate'] = df['winrate'].str[:-1].astype(float).div(100)
df['banrate'] = df['banrate'].str[:-1].astype(float).div(100)
```

Null Values

The only null row was with Udyr, so I had to manually enter his difficulty

```
df[df.isnull().any(axis='columns')]
```

✓ 0.0s

	name	winrate	banrate	averageKDA	difficulty	role	partype
141	Udyr	51.9%	8.1%	7.0/6.0/4.9	NaN	Top,Jungle	None

```
df.loc[141,'difficulty'] = 7
df.loc[141] # confirming changes
```

✓ 0.0s

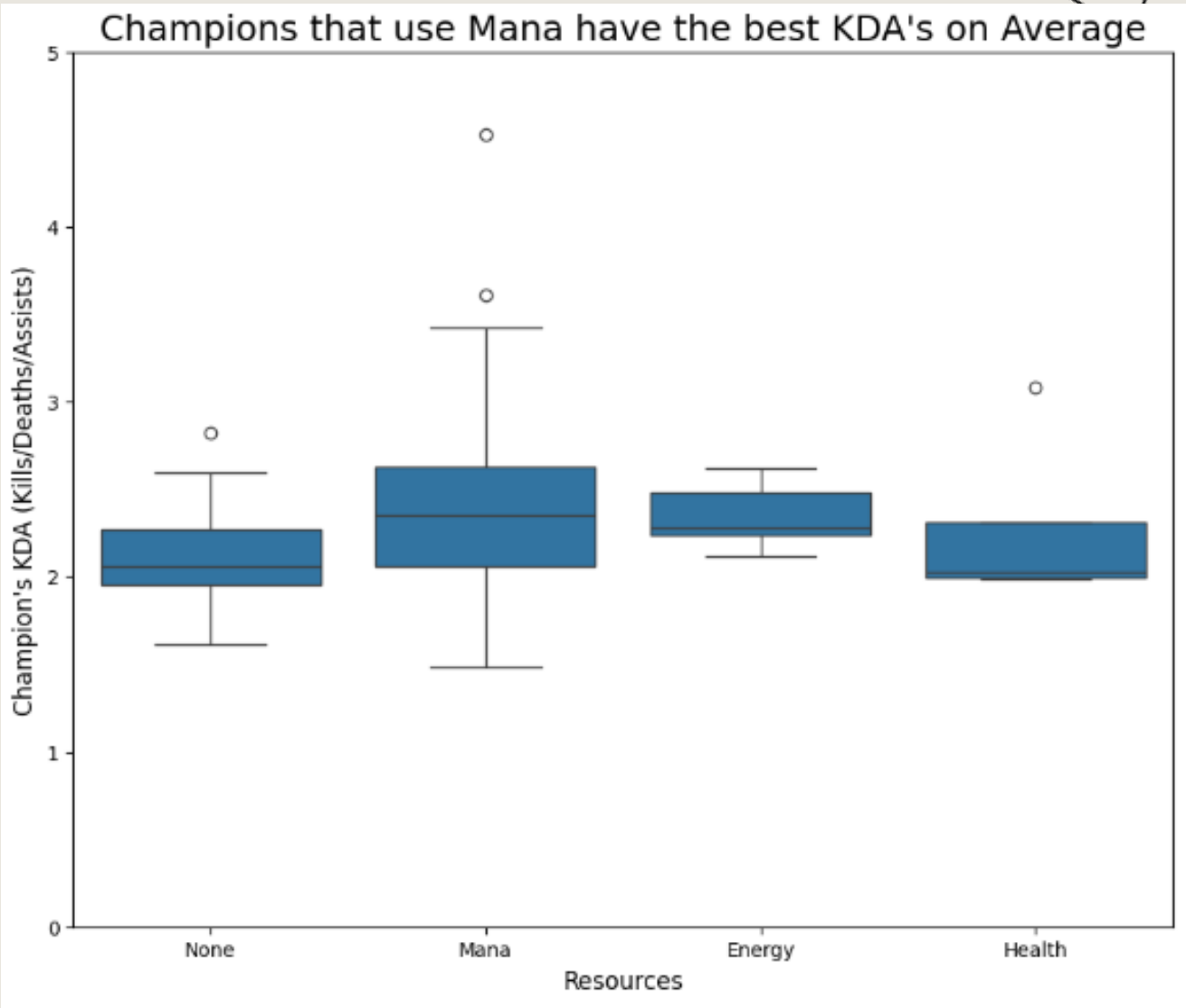
```
name                  Udyr
winrate              51.9%
banrate               8.1%
averageKDA     7.0/6.0/4.9
difficulty             7.0
role           Top,Jungle
partype                None
Name: 141, dtype: object
```

RESULTS

Now that the data has been cleaned, we can look for the right champions to answer our question

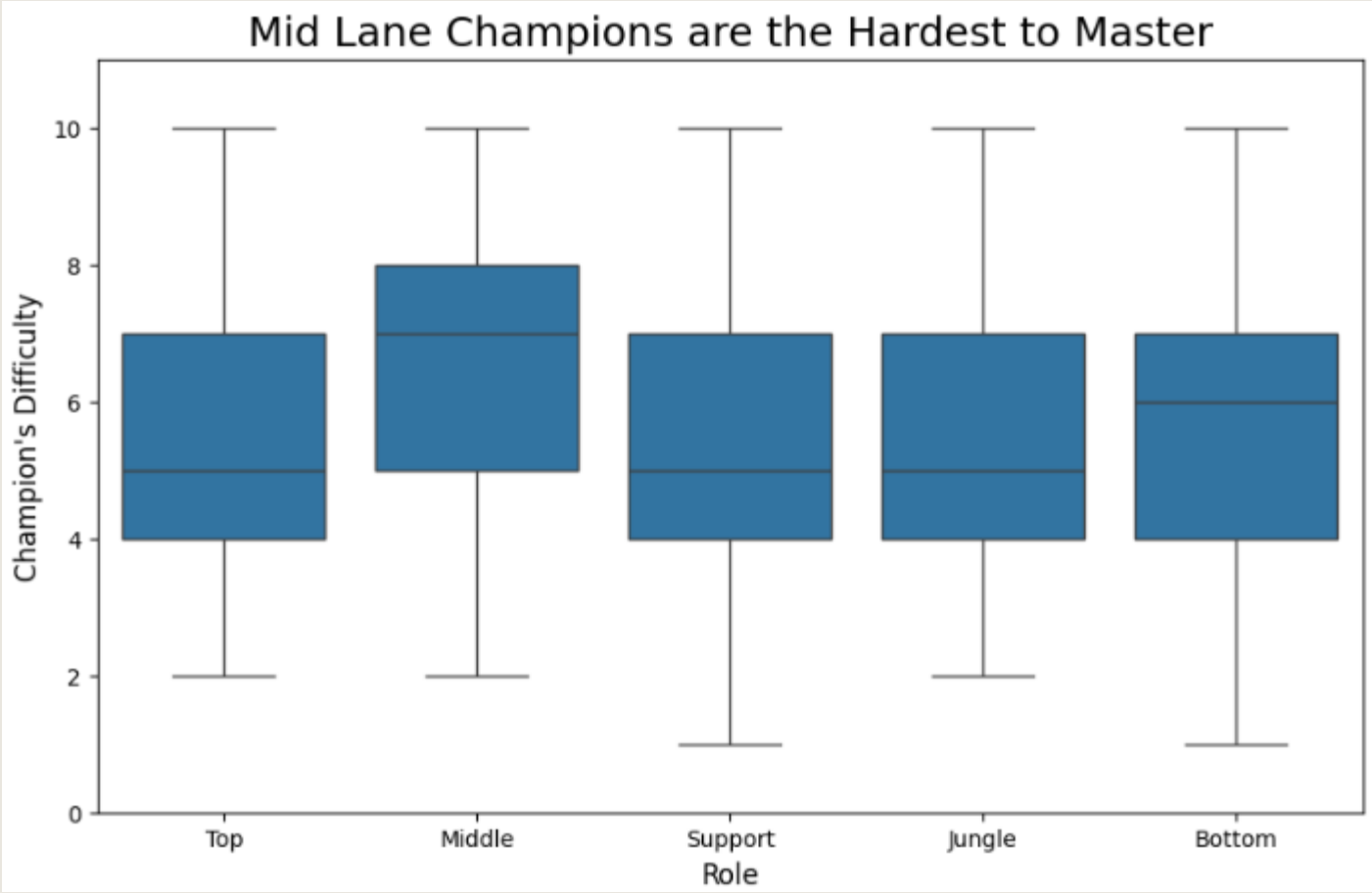
KDA vs Champion Resource

```
# creating a boxplot of kda vs resource type
plt.figure(figsize=(10,8))
sea.boxplot(data=df, x='partype', y ='averageKDA')
plt.title('Champions that use Mana have the best KDA\'s on Average', fontsize=18)
plt.xlabel('Resources', fontsize=12)
plt.ylabel('Champion\'s KDA (Kills/Deaths/Assists)', fontsize=12)
plt.ylim([0, 5])
plt.show()
```



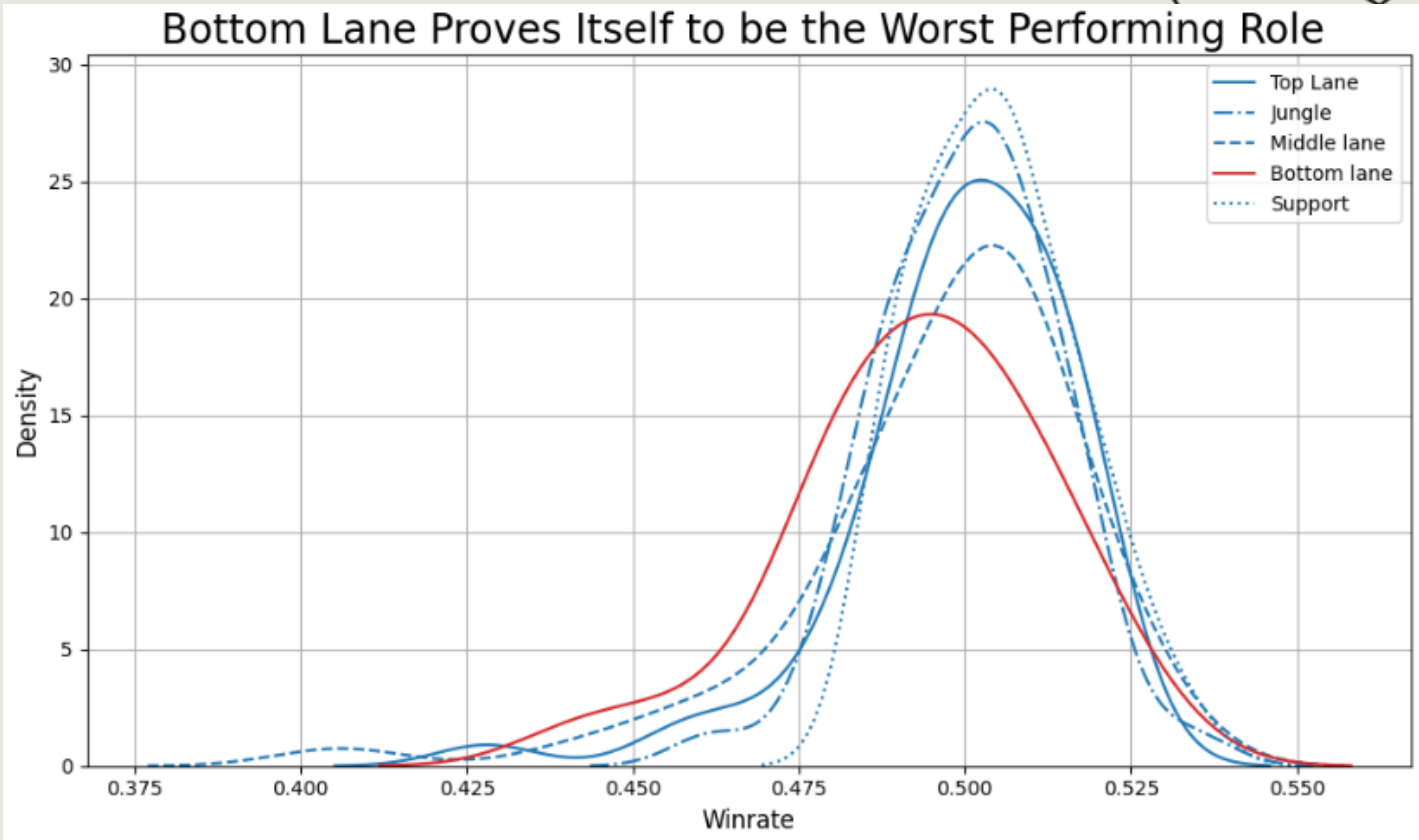
Difficulty vs Role

```
# exploding the dataframe to allow champs to be in more than one role
df_role_exploded = df.explode('role')

# creating a boxplot difficulty vs role
plt.figure(figsize=(10,6))
sea.boxplot(data=df_role_exploded, x='role', y ='difficulty')
plt.title('Mid Lane Champions are the Hardest to Master', fontsize=18)
plt.xlabel('Role', fontsize=12)
plt.ylabel('Champion\'s Difficulty', fontsize=12)
plt.ylim([0,11])
plt.show()
```



Winrates Per Role

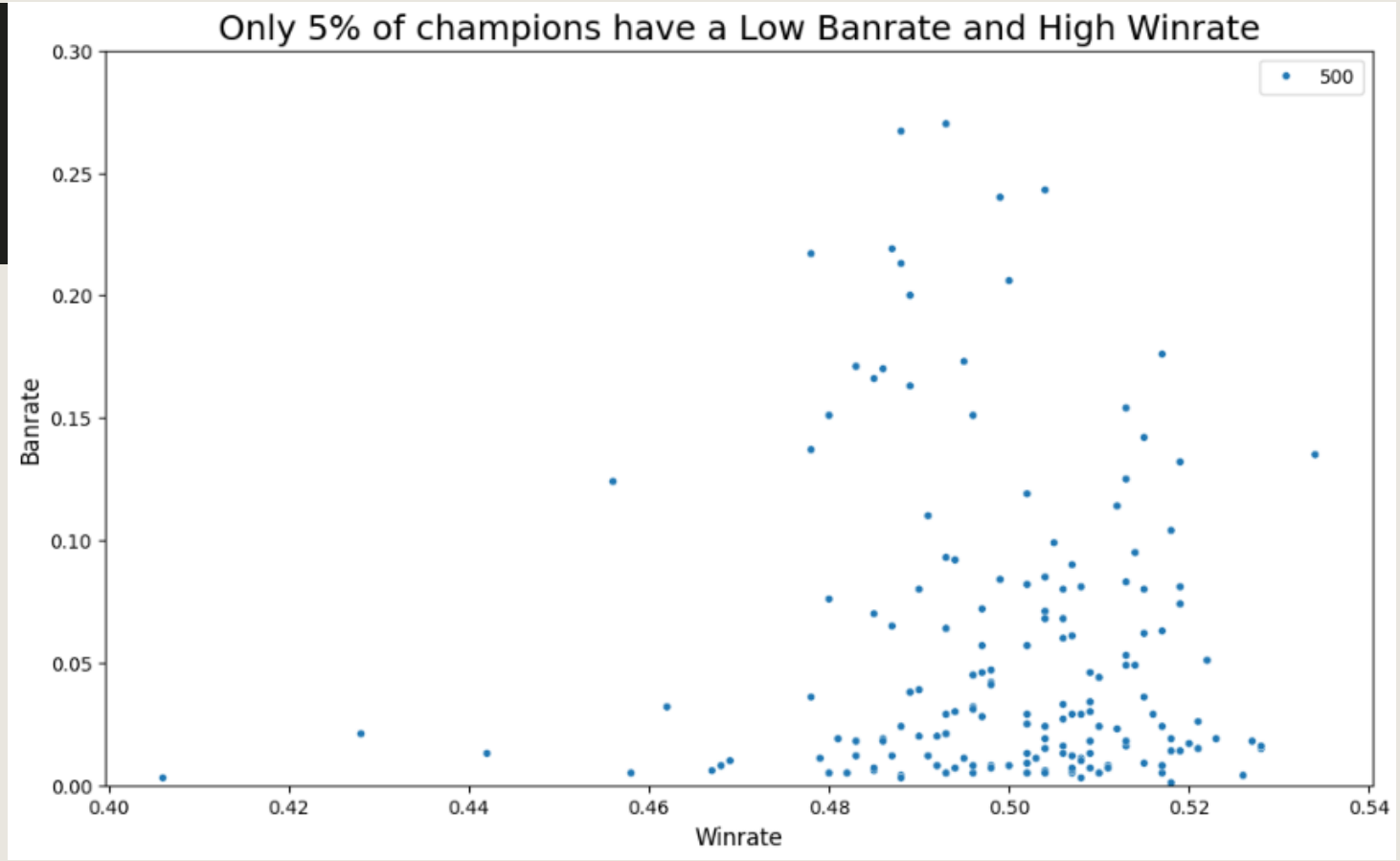
```
# creating separate dataframes for each role so we can test them individually
top_champs = df.loc[df['role'].apply(lambda row: 'Top' in row)]
jg_champs = df.loc[df['role'].apply(lambda row: 'Jungle' in row)]
mid_champs = df.loc[df['role'].apply(lambda row: 'Middle' in row)]
bot_champs = df.loc[df['role'].apply(lambda row: 'Bottom' in row)]
sup_champs = df.loc[df['role'].apply(lambda row: 'Support' in row)]

fig, ax = plt.subplots(figsize=(10,6))

sea.kdeplot(data=top_champs['winrate'], color='tab:blue')
sea.kdeplot(data=jg_champs['winrate'], color='tab:blue', linestyle= '-.')
sea.kdeplot(data=mid_champs['winrate'], color='tab:blue', linestyle='--')
sea.kdeplot(data=bot_champs['winrate'], color='tab:red')
sea.kdeplot(data=sup_champs['winrate'],color='tab:blue', linestyle=':')

plt.title('Bottom Lane Proves Itself to be the Worst Performing Role ', fontsize=20)
ax.legend(['Top Lane', 'Jungle', 'Middle lane', 'Bottom lane', 'Support'])
plt.xlabel('Winrate', fontsize=12)
plt.ylabel('Density', fontsize=12)
ax.grid()
plt.tight_layout()
plt.show()
```



Winrate vs Banrate per Champion

```
# creating a scatterplot of banrate vs winrate
plt.figure(figsize=[12,7])
winban_plot = sea.scatterplot(data=df, x='winrate', y='banrate', size=500)
plt.title('Very Few Champions have a Banrate', fontsize=18)
plt.xlabel('Winrate', fontsize=12)
plt.ylabel('Banrate', fontsize=12)
plt.ylim(0, 0.3)
plt.show()
```

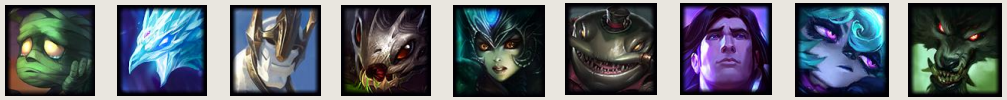
167 Champions to choose from

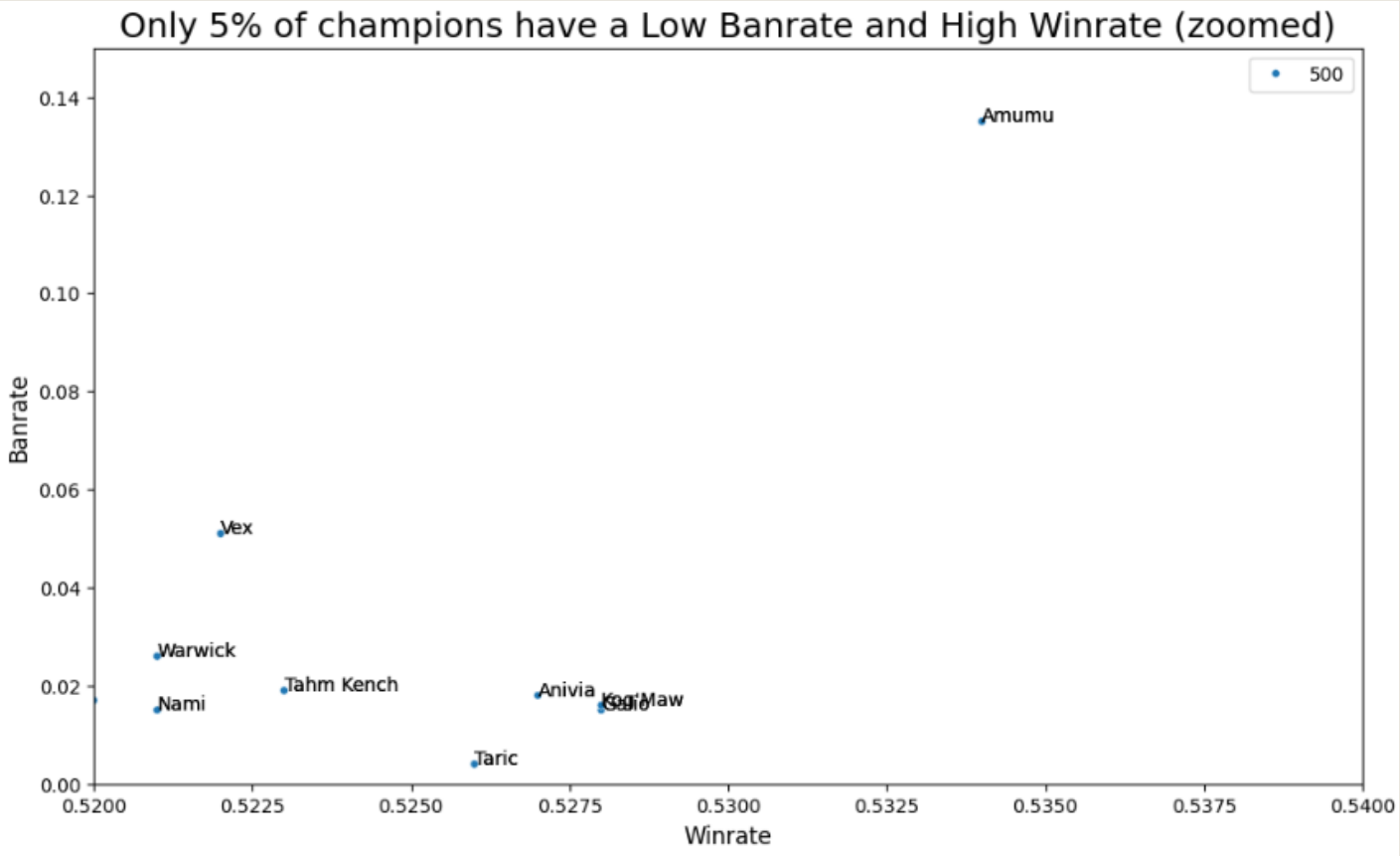


Winrate vs Banrate per Champion

```
# zooming in on the previous figure
winban_plot.axis([0.52, 0.54, 0, 0.15])
winban_champs = []
for x,y,z in zip(df['winrate'], df['banrate'], df['name']):
    if x > 0.52 and y < 0.15:
        winban_plot.text(x, y, z, horizontalalignment='left', size='medium', color='black')
        winban_champs.append(z) # storing those champions for later
winban_plot.set_title('More champions have an Abnormally Low Winrate than a High Winrate (zoomed in)', fontsize=18)
winban_plot.get_figure()
```

Now we have 9 champions to choose from.





CONCLUSION

Now that we have 9 champions to choose from, we can start filtering out the ones without the properties we like.

```
# create dataframe with those champions
best_champions = df.loc[df['name'].apply(lambda row: row in winban_champs)]
best_champions.head(len(best_champions))
```

✓ 0.0s

	name	winrate	banrate	averageKDA	difficulty	role	partype
5	Amumu	0.534	0.135	2.564516	3.0	[Jungle, Support]	Mana
6	Anivia	0.527	0.018	2.755102	10.0	[Middle]	Mana
35	Galio	0.528	0.015	2.763636	5.0	[Middle, Support]	Mana
66	Kog'Maw	0.528	0.016	2.268657	6.0	[Bottom]	Mana
86	Nami	0.521	0.015	1.909091	5.0	[Support]	Mana
130	Tahm Kench	0.523	0.019	2.460317	5.0	[Top, Support]	Mana
133	Taric	0.526	0.004	1.805970	3.0	[Support, Middle]	Mana
147	Vex	0.522	0.051	2.338710	2.0	[Middle]	Mana
153	Warwick	0.521	0.026	2.362069	3.0	[Jungle, Top]	Mana

After filtering out the bad properties, we're now left with 4 champions that you can choose from to help with your climb in League of Legends

```
# find the handful of champions that aren't in mid lane or bot lane
bad_champions = best_champions.loc[best_champions['role'].apply(lambda row: ('Middle' in row) or ('Bottom' in row))].index
final_champions = best_champions.drop(bad_champions)
final_champions.head(len(final_champions))
```

✓ 0.0s

	name	winrate	banrate	averageKDA	difficulty	role	partype
5	Amumu	0.534	0.135	2.564516	3.0	[Jungle, Support]	Mana
86	Nami	0.521	0.015	1.909091	5.0	[Support]	Mana
130	Tahm Kench	0.523	0.019	2.460317	5.0	[Top, Support]	Mana
153	Warwick	0.521	0.026	2.362069	3.0	[Jungle, Top]	Mana




THANK YOU