template

November 17, 2023

1 League of Legends Positions Impact Analysis

Name(s): Ahmed Mostafa and Ethan Vo

Website Link: https://ahmostafa147.github.io/dsc80-project/

1.0.1 Which player position, when achieving more kills than their counterpart on the opposing team, has the greatest impact on boosting the overall win rate?

1.1 Code

[1]:
```
import pandas as pd
import numpy as np
import os

import plotly.express as px
pd.options.plotting.backend = 'plotly'
```

1.1.1 Cleaning and EDA

We extracted the columns `gameid`, `side`, `position`, and `kills` from the League of Legends DataFrame as they are relevant to our proposed question. We cleaned the data by removing the rows that contain information about the team. We did this by removing the rows that had "team" value in the `position` column. We then added a new column to the DataFrame that identifies each row as whether it belongs to a position that had more kills than its counterpart in the opposing team.

[2]:
```
lol_data = pd.read_csv('lol_data.csv', usecols=["gameid", "datacompleteness",␣
↪"side", "position", "kills", "teamkills", "result", "league", "killsat15"])
lol_data = lol_data.query("position != 'team'")
red = lol_data.sort_values(by=["gameid", "position"]).query("side == 'Red'")
blue = lol_data.sort_values(by=["gameid", "position"]).query("side == 'Blue'")
red["has_more_kills"] = np.array(red['kills']) > np.array(blue['kills'])
blue["has_more_kills"] = np.array(red['kills']) < np.array(blue['kills'])
col_added = red.merge(blue, how='outer').sort_values(by=["gameid", "position"])
```

1.1.2 Univariate Charts

In order to increase our awareness of the data we have, we produced an interactive pie chart using `plotly` that reprensents the percentage of players who had more kills and won vs players who had more kills but won. This visualization is helpful to give us insight of what to expect from our proportions that represents the impact of having more kills. As we can see, 79.4% of the players who had more kills won. This gives us insight about what our proportions would look like per position. It makes us expect that having more kills should have a big impact on your chances of winning.

[3]:
```
more_kills_won = (col_added
                  .query("result == 1 and has_more_kills == True")
                  .shape[0] /
                  col_added
                  .query("has_more_kills == True")
                  .shape[0]
                 )
more_kills_lost = 1 - more_kills_won
data = {'rate': [more_kills_won, more_kills_lost],
        'Label': ['Had more kills and won', 'Had more kills but lost']}
df = pd.DataFrame(data)
fig = px.pie(df, values='rate', names='Label', title='Players winning rate when␣
 ↪having more kills')
fig.show()
```

This histogram shows the distribution of kills counts per position for winning vs losing players. …

[4]:
```
fig = (px.histogram(lol_data,
                    x="kills",
                    facet_col= "position",
                    facet_row = "result",
                    title="Kills Distribution Per Position",
                    histnorm="probability density")
      )
fig.show()
```

1.1.3 Bivariate Chart

[5]:
```
prop_per_position_more = (col_added
                     .query("has_more_kills == True and result == 1")
                     .groupby("position")
                     .size() /
                     col_added
                     .groupby('position')
                     .size()
                    )
prop_per_position_more.name = "Proportions"
fig2 = px.bar(prop_per_position_more, title="Win Rate of Position with more␣
 ↪Kills")
fig2.update_layout(
    legend=dict(title='Variable'),  # Add legend title
    yaxis_title='Win Rate', # Add y-axis label
```

```
)
fig2.show()
```

[6]:
```
prop_per_position_less = (col_added
                          .query("has_more_kills == False and result == 1")
                          .groupby("position")
                          .size() /
                          col_added
                          .groupby('position')
                          .size()
                         )
observed_stat = (prop_per_position_more - prop_per_position_less)
observed_stat.name = "Proportions"
fig2 = px.bar(observed_stat, title="Increase in Win Rate of Position With More␣
 ↪Kills vs Less Kills")
fig2.update_layout(
    legend=dict(title='Variable'),  # Add legend title
    yaxis_title='Win Rate', # Add y-axis label
)
fig2.show()
```

Interesting Aggregates

[7]:
```
pivot_table = lol_data[['result', 'kills', 'position']].groupby(["result",␣
 ↪'position']).mean().reset_index().pivot_table(index="result",␣
 ↪columns="position", values="kills")
pivot_table
```

[7]:

position	bot	jng	mid	sup	top
result					
0	2.723914	1.801244	2.271909	0.582250	1.703583
1	6.277571	3.468065	4.851560	1.016524	3.512904

1.1.4 Assessment of Missingness

[8]:
```
missing_assessment_data = lol_data[["killsat15", "league"]]
missing_assessment_data = missing_assessment_data.assign(missing =␣
 ↪missing_assessment_data["killsat15"].isna())
observed_pivot_table = (
        missing_assessment_data
        .pivot_table(index='league', columns='missing', aggfunc='size')
        .apply(lambda x: x / x.sum())
    )
observed_pivot_table.fillna(0, inplace=True)
observed_tvd = observed_pivot_table.diff(axis=1).iloc[:, -1].abs().sum() / 2
fig = observed_pivot_table.plot(kind='bar', title='League by Missingness of␣
 ↪Killsat15 Values', barmode='group')
fig.show()
```

```
observed_pivot_table
```

[8]:

missing	False	True
league		
AL	0.018911	0.000000
CBLOL	0.026454	0.000000
CBLOLA	0.027110	0.000000
CDF	0.007433	0.000000
CT	0.004700	0.000000
DDH	0.009401	0.000000
EBL	0.017381	0.000000
EL	0.004482	0.000000
EM	0.029624	0.000000
EPL	0.009401	0.000000
ESLOL	0.033013	0.000000
GL	0.009729	0.000000
GLL	0.017927	0.000000
HC	0.015413	0.000000
HM	0.017709	0.000000
IC	0.007324	0.000000
LAS	0.027547	0.000000
LCK	0.053236	0.000000
LCKC	0.055203	0.000000
LCO	0.015304	0.000000
LCS	0.028859	0.000000
LDL	0.000000	0.527094
LEC	0.031373	0.000000
LFL	0.026454	0.000000
LFL2	0.027219	0.000000
LHE	0.006449	0.000000
LJL	0.028203	0.000000
LJLA	0.010057	0.000000
LLA	0.020988	0.000000
LMF	0.009401	0.000000
LPL	0.000000	0.464901
LPLOL	0.017490	0.000000
LRN	0.012680	0.000000
LRS	0.013008	0.000000
LVP SL	0.026891	0.000000
MSI	0.008308	0.000000
NACL	0.094775	0.000000
NEXO	0.018474	0.000000
NLC	0.017162	0.000000
PCS	0.032029	0.000000
PGN	0.021972	0.000000
PRM	0.026454	0.000000
SL (LATAM)	0.009838	0.000000

```
TCL          0.019676   0.000000
UL           0.026782   0.000000
VCS          0.035308   0.000000
VL           0.009620   0.000000
WLDs         0.013227   0.008005
```

1.1.5 Permutation Testing (killsat15 column depends on league)

[9]:
```
n_repetitions = 500
shuffled = lol_data[["killsat15", "league"]]
shuffled = shuffled.assign(missing = shuffled["killsat15"].isna())

tvds = []
for _ in range(n_repetitions):

    shuffled['league'] = np.random.permutation(shuffled['league'])

    # Computing and storing the TVD.
    pivoted = (
        shuffled
        .pivot_table(index='league', columns='missing', aggfunc='size')
        .apply(lambda x: x / x.sum())
    )
    pivoted.fillna(0, inplace=True)
    tvd = pivoted.diff(axis=1).iloc[:, -1].abs().sum() / 2
    tvds.append(tvd)
```

[10]:
```
fig = px.histogram(pd.DataFrame(tvds), histnorm='probability',
                   title='Empirical Distribution of the TVD')
fig.add_vline(x=observed_tvd, line_color='red')
fig.add_annotation(text=f'<span style="color:red">Observed TVD =␣
 ↪{round(observed_tvd, 4)}</span>',
                   x=1,showarrow=False, y=0.10)
pval = np.mean(np.array(tvds) >= observed_tvd)
fig.show()
print("P-value: " + str(pval))
```

```
P-value: 0.0
```

1.1.6 Permutation Testing (killsat15 column doesn't depend on side)

[11]:
```
missing_assessment = lol_data[["side", "killsat15"]]
missing_assessment = missing_assessment.assign(missing =␣
 ↪missing_assessment['killsat15'].isna())
missing_assessment_pivoted = (
        missing_assessment
        .pivot_table(index='side', columns='missing', aggfunc='size')
```

```
        .apply(lambda x: x / x.sum())
    )
missing_assessment_pivoted.fillna(0, inplace=True)
observed_tvd2 = missing_assessment_pivoted.diff(axis=1).iloc[:, -1].abs().sum()␣
 ↪/ 2
fig3 = missing_assessment_pivoted.plot(kind='bar', title='Side by Missingness␣
 ↪of Killsat15 Values', barmode='group')
fig3.show()
missing_assessment_pivoted
```

[11]:

```
missing  False  True
side
Blue       0.5    0.5
Red        0.5    0.5
```

[12]:

```
n_repetitions = 500
shuffled = lol_data[["side", "killsat15"]]
shuffled = shuffled.assign(missing = shuffled["killsat15"].isna())

tvds = []
for _ in range(n_repetitions):

    shuffled['side'] = np.random.permutation(shuffled['side'])

    # Computing and storing the TVD.
    pivoted = (
        shuffled
        .pivot_table(index='side', columns='missing', aggfunc='size')
        .apply(lambda x: x / x.sum())
    )
    pivoted.fillna(0, inplace=True)
    tvd = pivoted.diff(axis=1).iloc[:, -1].abs().sum() / 2
    tvds.append(tvd)
```

[13]:

```
fig4 = px.histogram(pd.DataFrame(tvds), histnorm='probability',
                   title='Empirical Distribution of the TVD')
fig4.add_vline(x=observed_tvd2, line_color='red')
fig4.add_annotation(text=f'<span style="color:red">Observed TVD =␣
 ↪{round(observed_tvd2, 4)}</span>',
                   x=0.05,showarrow=False, y=0.15)
pval = np.mean(np.array(tvds) >= observed_tvd2)
fig4.show()
pval
```

[13]:

```
1.0
```

1.1.7 Hypothesis Testing

- **Null Hypothesis**: The proportion of **support** position winning and having higher kills is equal to the proportion of support position winning and having less kills.
- **Alternative Hypothesis**: The proportion of support position winning and having higher kills is less than the proportion of support position winning and having less kills.

[14]:
```
samples = (np.random.multinomial(len(lol_data['gameid'].unique()),
                                pvals=[0.5,0.5],
                                size = 100000)
           / 10772 # Converts the samples to proportions
           / 2 # divides in half since the overall population (winning support)␣
 ↪is 0.5,
               # so proportions should sum up to 0.5
          )
pval = sum((samples[:, 0] - samples[:, 1]) <= observed_stat["sup"])
```

[15]:
```
difference_in_proportions = pd.Series(samples[:, 0] - samples[:, 1])
difference_in_proportions.name = "Difference in Proportions"
```

[16]:
```
fig = px.histogram(
    difference_in_proportions,
    histnorm='probability',
    title='Empirical Distribution of the Differences in Proportions Between \
Winning Support with Higher Kills vs with Lower Kills')
fig.add_vline(x = observed_stat["sup"], line_color='red')
print("P-value: " + str(pval))
fig.show()
```

```
P-value: 0
```