Data 301 Final Project

Josiah Lashley and Steven Taruc

League of Legends

League of Legends

League of Legends

League of Legends

League of Legends

Champions

League of Legends

League of Legends

Match Analysis Overview

Exploratory data analysis and summary statistics

Machine Learning on predicting game outcome

Map location analysis

Player Attributes

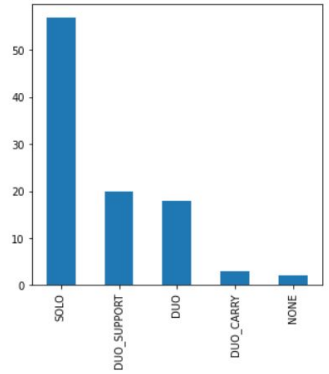
	platformId	gameId	champion	queue	season	timestamp	role	lane
0	NA1	3328664670	429	420	13	1584501608128	SOLO	TOP
1	NA1	3328526992	104	420	13	1584498834816	DUO	MID
2	NA1	3327643642	104	420	13	1584420800770	SOLO	TOP
3	NA1	3327572194	429	420	13	1584417162854	SOLO	TOP
4	NA1	3327515909	429	420	13	1584415337824	SOLO	TOP

Player Attributes

What role do you play the most?

```
role_counts = df_games['role'].value_counts()
most_used_role = role_counts[[0]].index[0]
role_counts, role_counts.plot.bar()
```

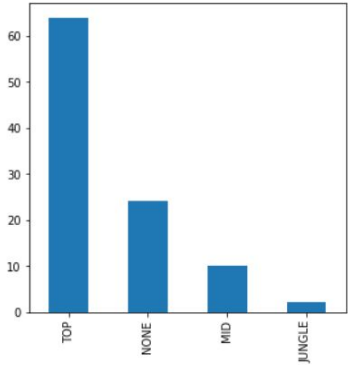
```
(SOLO            57
 DUO_SUPPORT     20
 DUO             18
 DUO_CARRY        3
 NONE             2
 Name: role, dtype: int64,
 <matplotlib.axes._subplots.AxesSubplot at 0x7fe8fc8669b0>)
```



What lane do you play the most?

```
lane_counts = df_games['lane'].value_counts()
most_used_lane = lane_counts[[0]].index[0]
lane_counts, lane_counts.plot.bar()
```

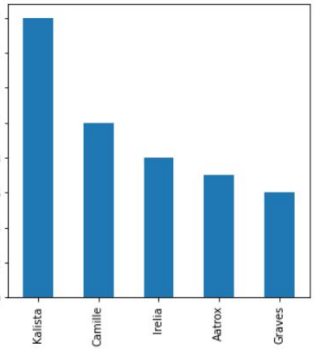
```
(TOP        64
 NONE       24
 MID        10
 JUNGLE      2
 Name: lane, dtype: int64,
 <matplotlib.axes._subplots.AxesSubplot at 0x7fe8fc7ba320>)
```



What champion do you play the most?

```
champ_counts = df_games['champion_name'].value_counts()
most_used_champ = champ_counts[[0]].index[0]
champ_counts[:5], champ_counts[:5].plot.bar()
```

```
(Kalista    16
 Camille    10
 Irelia      8
 Aatrox      7
 Graves      6
 Name: champion_name, dtype: int64,
 <matplotlib.axes._subplots.AxesSubplot at 0x7fe8fc6f7b00>)
```

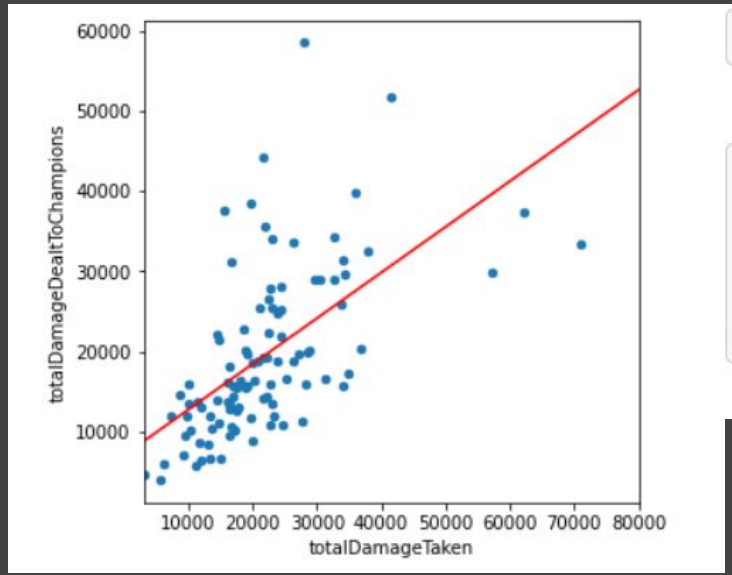


Match Statistics

	win	item0	item1	item2	item3	item4	item5	item6	kills	deaths	assists	largestKillingSpree	largestMultiKill	killingSpr
0	False	3153	3085	3006	1053	1055	0	3340	4	7	3	2	1	2
1	False	2055	3046	3065	3047	3072	3812	3340	7	9	10	3	2	2
2	False	2055	3812	3047	3123	1055	1038	3340	2	12	3	0	1	0
3	True	3153	3033	1055	3085	3006	1083	3340	2	8	8	0	1	0
4	False	3153	1053	3006	1055	1038	3085	3340	3	8	1	2	1	1

Machine Learning

Simple Linear Regression on damage to others vs. damage taken




```
model.score(X=X_train,y=y_train)
```

```
0.3530405561675627
```

```
scores = -cross_val_score(model,
                          X=X_train,
                          y=y_train,
                          scoring="neg_mean_squared_error",
                          cv=10)
np.sqrt(scores.mean())
```

```
8668.7789915207
```

Machine Learning

Predicting Win Probability using Pre-match Attributes

We are able to setup a k-nearest neighbors classifier that predicts game outcome from pre-match attributes.
Using our most used champion, role, and lane from our collected data of 100 matches, we predict to have a 60% chance of winning and 40% chance of losing a match.

```
test1 = pd.DataFrame({
    'champion_name':[most_used_champ],
    'role':[most_used_role],
    'lane':[most_used_lane]
})
test1
```

	champion_name	role	lane
0	Kalista	SOLO	TOP

```
pipeline.predict_proba(test1)
```

```
array([[0.6, 0.4]])
```

Timeline Data

Each time an event happens

- Event type
- Timestamp
- Location
- Killers, victims, etc

Gbay's Kills vs My Kills

- Before and After 17 mins

Machine Learning Model

Ran a MLP Classification Model to predict the outcome of the game

- Only looked at data past 10 min
  a. Really hard to predict outcomes based before 10 mins
- Each time an event happened, what was your status in the game at that time
  a. Kills, deaths, assists, teamwork score, tower score

Gbay's Win Predictions

```
features = ["tower_assisting", "tower_kill", "kill", "death","level_up","assist",
            "minute","kills","deaths","level","kill_participation","tower_kills",
            "team_participation"]

model1 = make_pipeline(
  StandardScaler(with_mean=False),
  #KNeighborsClassifier(n_neighbors=1, metric="manhattan")
  #RandomForestRegressor(n_estimators=100,oob_score=True)
  MLPClassifier(hidden_layer_sizes=(10,), max_iter=10000)
  )

model1.fit(X=df_gbay_10[features],y=df_gbay_10["win"])

cv_errs = cross_val_score(model1,X=df_gbay_10[features],
                                 y=df_gbay_10["win"],
                                 scoring="f1_macro",
                                 #scoring="neg_mean_squared_error",
                                 cv=30)
cv_errs.mean()
```

```
0.7298038924926129
```

My Win Predictions

```
features = ["tower_assisting", "tower_kill", "kill", "death","level_up","assist",
            "minute","kills","deaths","level","kill_participation","tower_kills",
            "team_participation"]

model2 = make_pipeline(
  StandardScaler(with_mean=False),
  #KNeighborsClassifier(n_neighbors=1, metric="manhattan")
  #RandomForestRegressor(n_estimators=100,oob_score=True)
  MLPClassifier(hidden_layer_sizes=(10,), max_iter=10000)
  )

model2.fit(X=df_me_10[features],y=df_me_10["win"])

cv_errs = cross_val_score(model2,X=df_me_10[features],
                                 y=df_me_10["win"],
                                 scoring="f1_macro",
                                 #scoring="neg_mean_squared_error",
                                 cv=30)
cv_errs.mean()
```

```
0.6892535323406421
```


LEAGUE OF LEGENDS
Thank You!!