Predicting the Outcomes of Matches in League of Legends

Justin Tsao

Data Science Capstone Project, March 2019

Disclaimer

- *This project isn't endorsed by Riot Games and doesn't reflect the views or opinions of Riot Games or anyone officially involved in producing or managing League of Legends. League of Legends and Riot Games are trademarks or registered trademarks of Riot Games, Inc. League of Legends © Riot Games, Inc.*

League of Legends and Esports

An icon in the Esports Industry

Two teams battle 5v5 in real time

Played professionally around the world

Endless combination of champions, spells, items, runes, and strategies

Big sponsors and investors – Samsung, Rocket Mortgage, Mark Cuban…

Top players earn 6 figure salaries + prizes

The World Championship in 2013 held in LA's Staples Center sold out in an **HOUR!** Tickets sold for *100*s of dollars each.

Problem to solve

Can the outcome of competitive or ranked matches be predicted?

What factors/features to use?

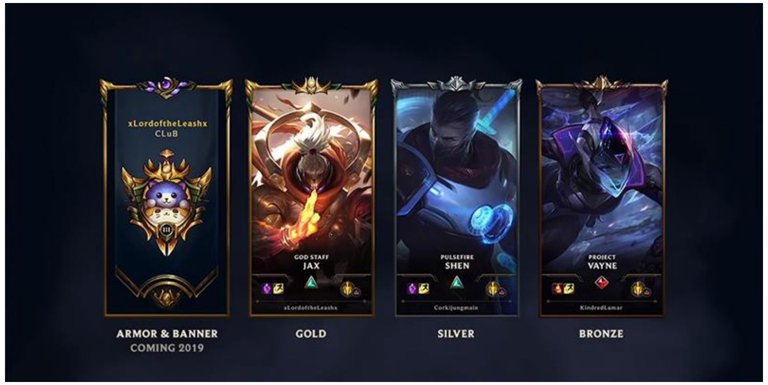
Classification: WIN OR LOSE

Who Can Benefit?

Professional Esports Teams

Riot Games

Ranked and High Skill Players



Benefits

- Intuition and insight on team compositions
- Talking point for shout casting
- Improving strategies
- For fun!

Acquiring the Data

- Riot Games API
- Match data from *NA ranked games
- Diamond Tier and above
- Pulled ~146K matches

*NA – North America

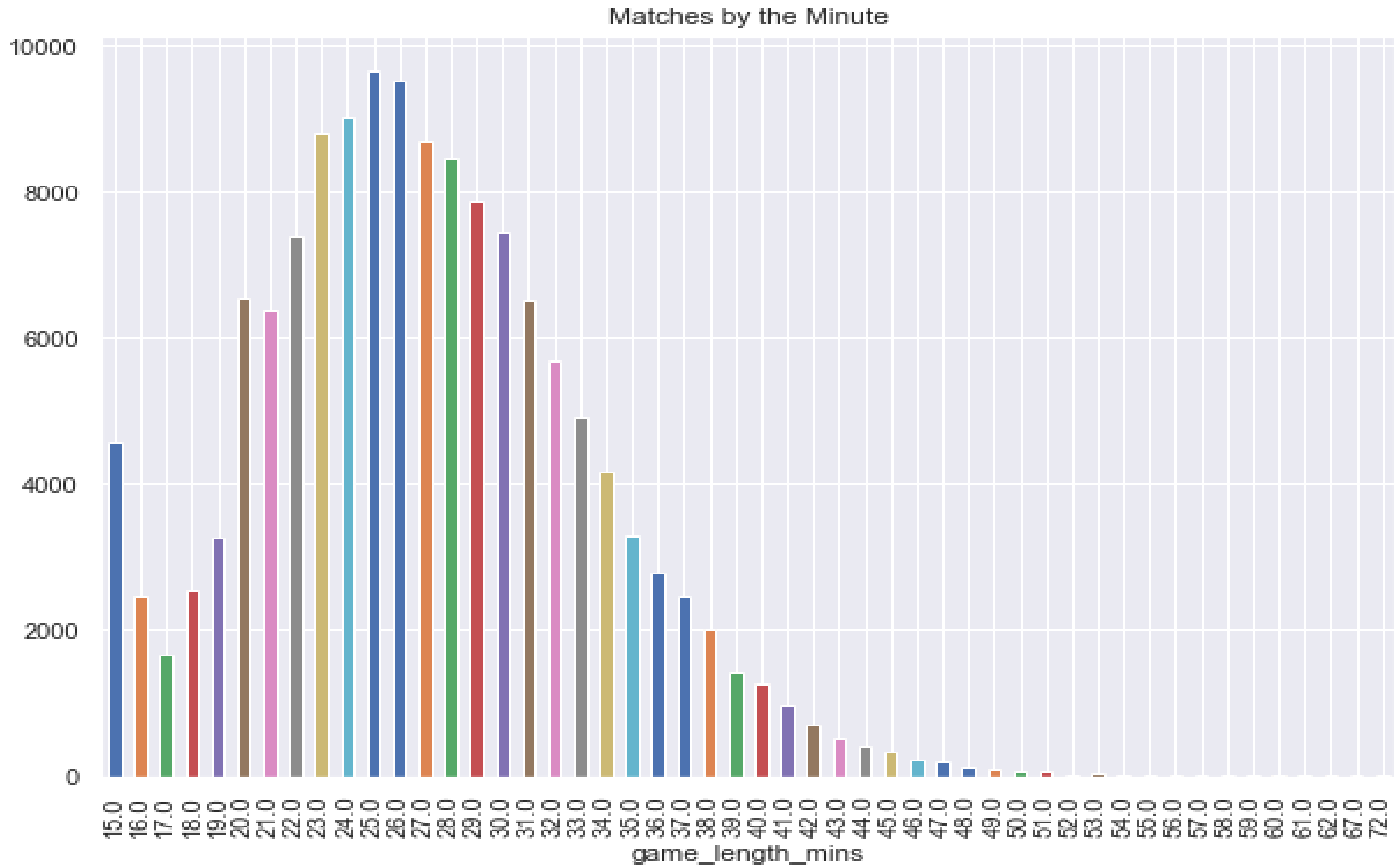
Data Wrangling using Python

- API Rate limiting -> 50+ hours to obtain data
- Parsing JSON -> convert to .csv and Pandas Data Frame
- Map feature id numbers to strings
- No missing values!
- Keep outliers
- Removed useless features
- Create new features from the data

Explore Data Visually

Inferential Statistics

Exploratory Data Analysis



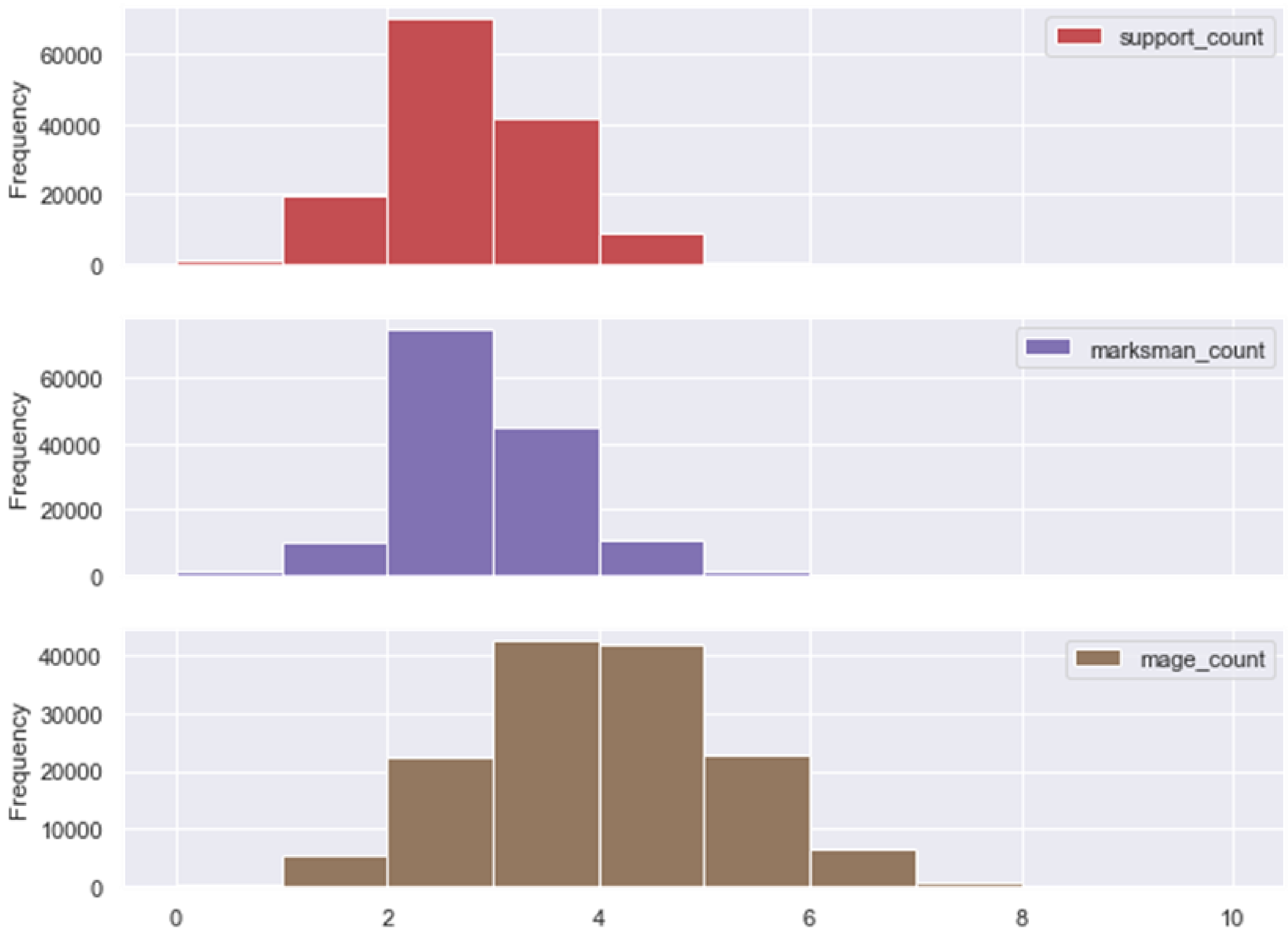
Matches under 15 mins were removed form the data set because they represented early surrenders or player disconnects.

count	142773.000000
mean	27.471626
std	6.372879
min	15.016667
25%	23.100000
50%	26.950000
75%	31.400000
max	72.966667

Name: game_length_mins, dtype: float64

EDA – Match Times

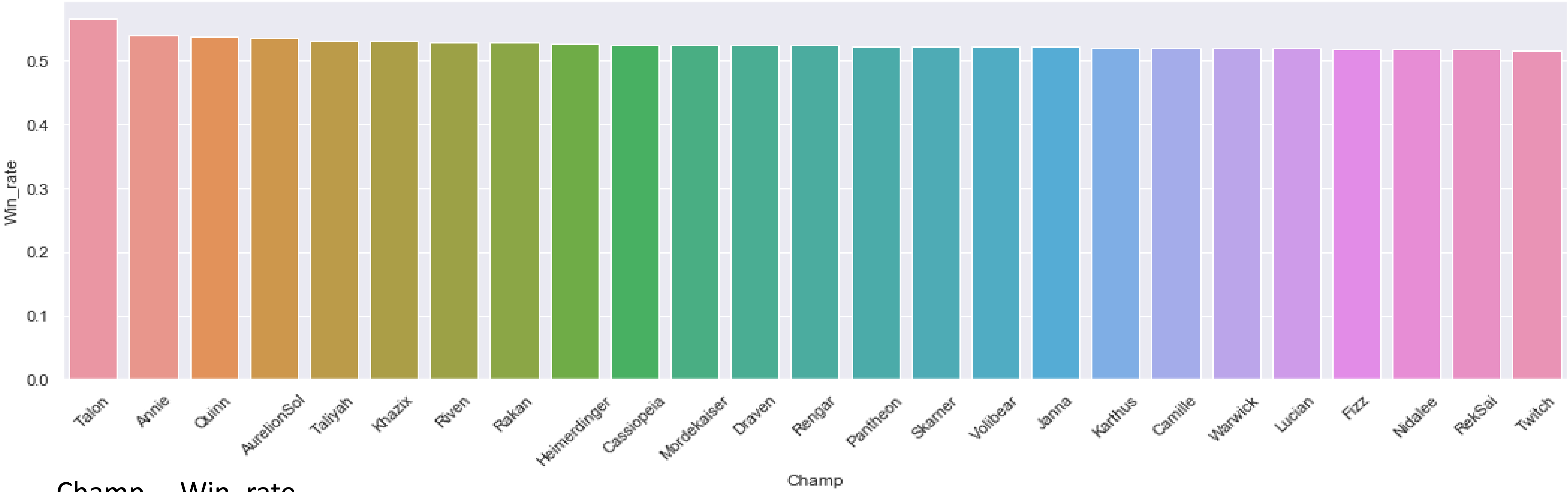
EDA – Champion Roles



supports: 325173, 12.81%
marksmen: 343258, 13.52%
mage: 504429, 19.87%
tank: 326048, 12.84%
assassin: 419309, 16.52%
fighter: 619549, 24.41%

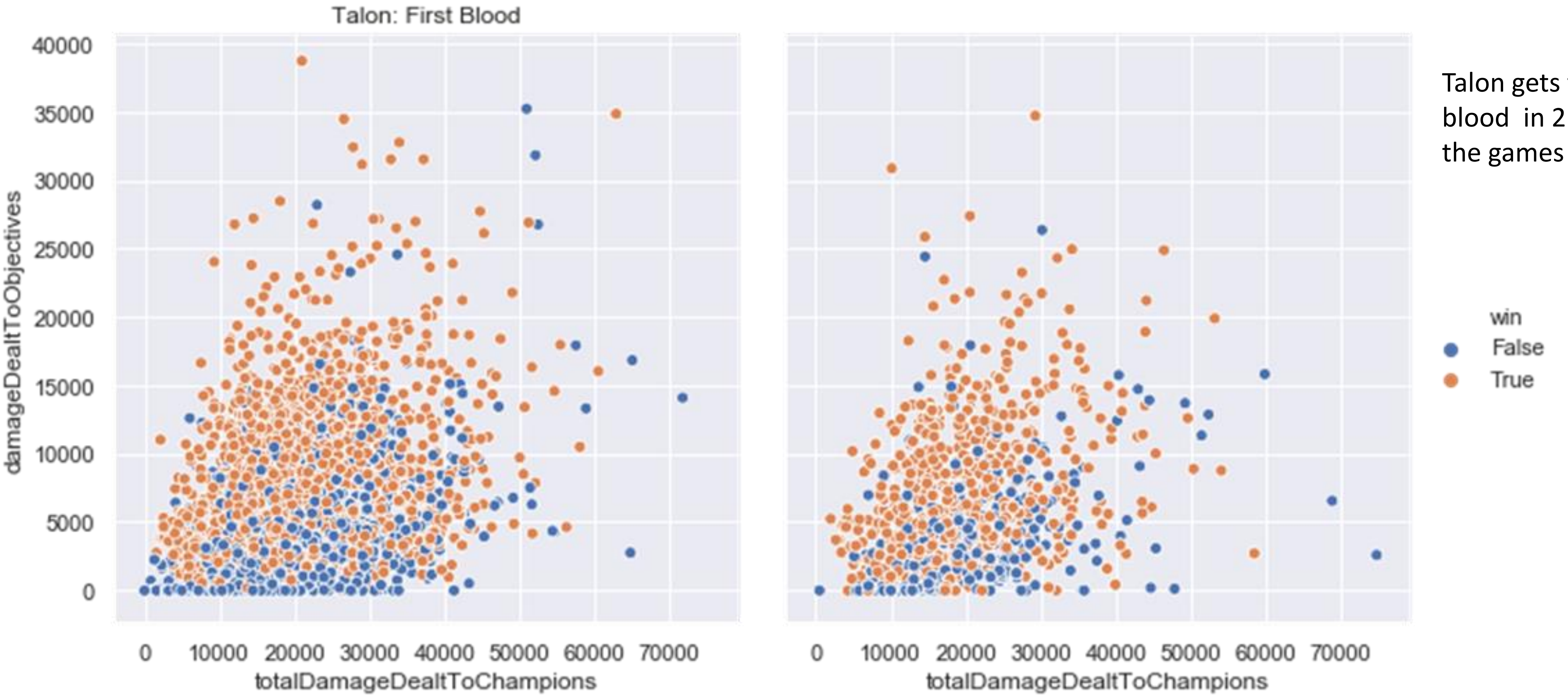
The count and percentage of the champions played from the data.

EDA -Top 25 Champions by Win Rate



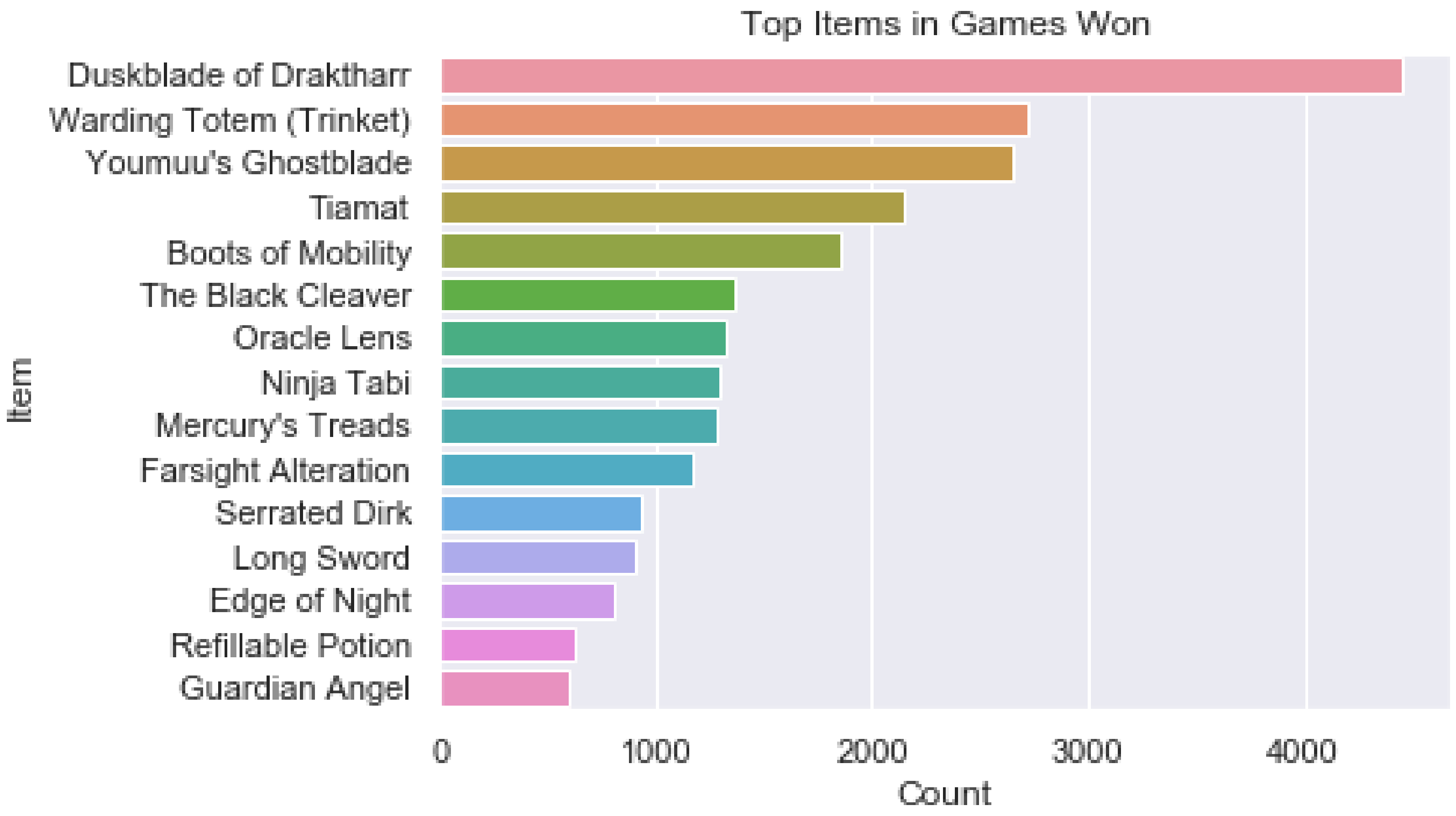
Champ	Win_rate
Talon	0.565446
Annie	0.540104
Quinn	0.536932
Aurelion	0.536824
Taliyah	0.532367

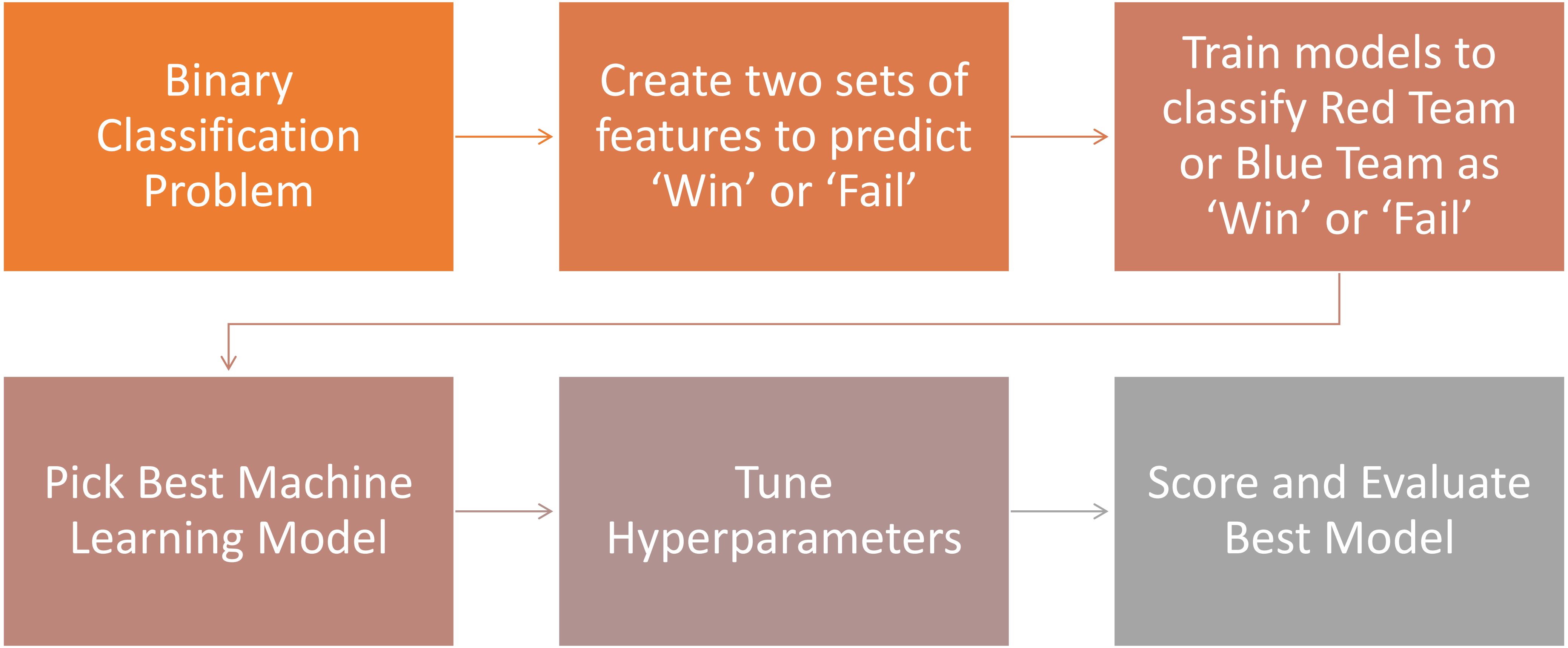
EDA - Champion Talon First Blood



Talon gets first blood in 21.93% of the games played.

EDA – Talon Top Items in Games Won





Supervised Machine Learning

Choosing Features

- Based on player pre-match decisions
- Champion Picks
- Summoner Spells
- Runes

Intuition on Feature Selection

Highly skilled players assumed to have in depth game knowledge

Players have direct control on these features

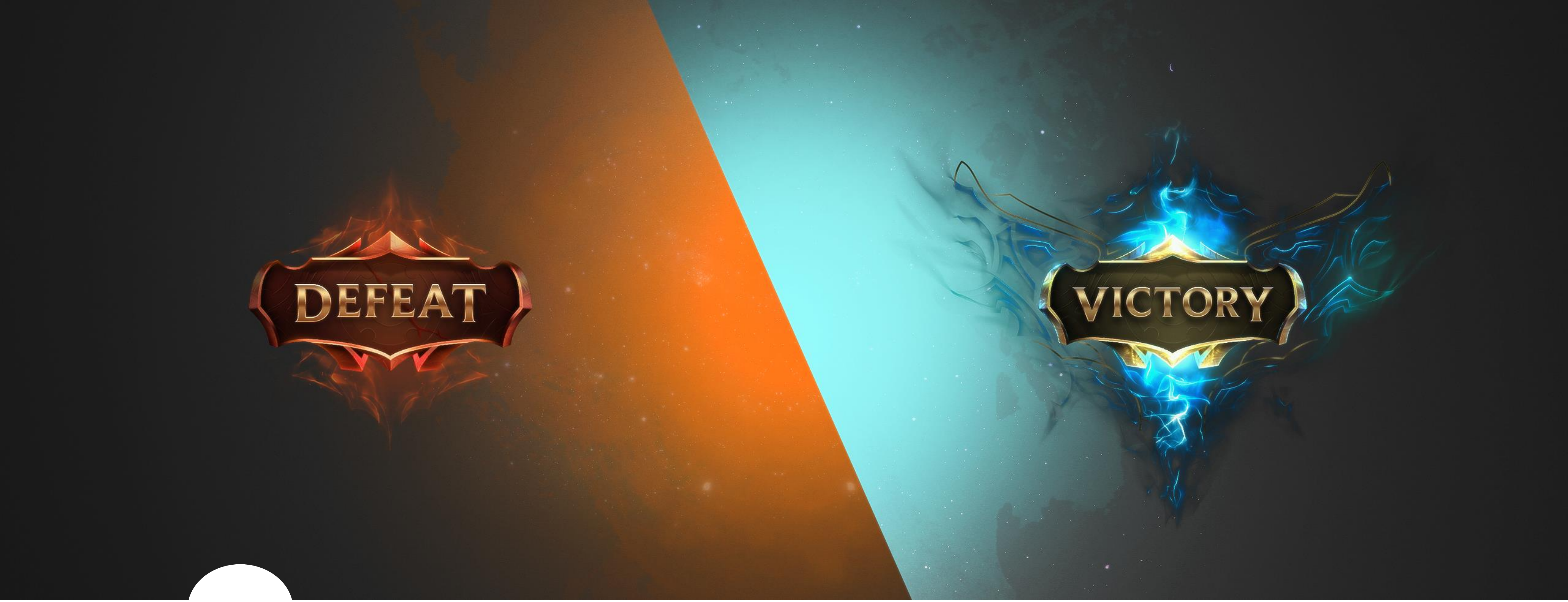
At high ranked tiers, everyone is around the same skill level

- Therefore, team composition + spells/runes should be good predictors

Testing Two Feature Sets

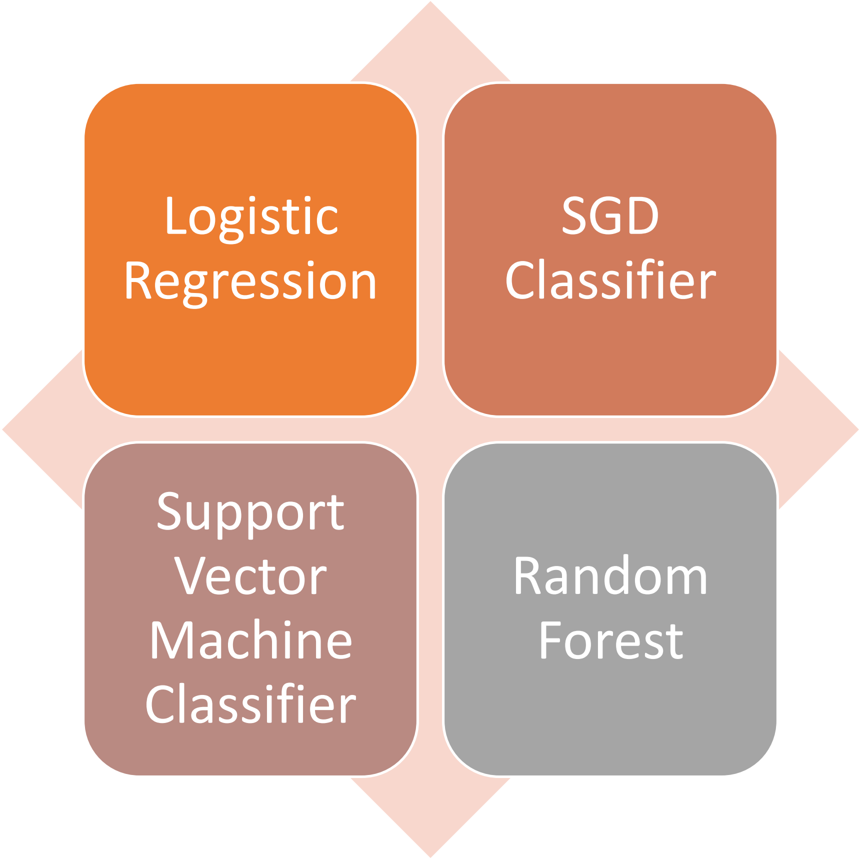
- Feature Set I
  - Champion Picks
  - Summoner Spells
  - Primary Runes
  - Secondary runes
  - Primary Role of Champion
  - Secondary Role of Champion
  - Role Counts

- Feature Set II
  - Champion Picks
  - Summoner Spells
  - Primary Runes
  - Secondary Runes
  - Removed features related to champion roles – may just add noise



Target Values

- Can either be Red Side Win or Blue Side Win
- Both can be classified as a ‘Win’ or ‘Fail'



Choosing a Classification Model

Out of Box Accuracy Scores for Models

Random Forest – 0.81

Logistic Regression – 0.57

SGD Classifier – 0.54

Support Vector Machine – 0.51

Best Classifier?

Random Forest

- Evaluating Models
  - Hyper parameter tuning
  - Classification report
  - Cross Validation
  - ROC Curves
  - AUC score

To see full model testing:
https://github.com/jltsao88/Capstone_Project_1/blob/master/Machine_Learning.ipynb

Classification Report

Performance measures for the model

- Precision – the proportion of positive identifications that were actually correct
- Recall – the proportion of actual positive identifications and were identified correctly
- F1-score – a measure of a test's accuracy which considers both recall and precision

*See appendix at end for more details on above metrics

Random Forest Model – Red Side Win

No Tuning of Hyper Parameters
Feature Set I
Set test size: 20%
5-Fold Cross Validation, Accuracy Score: 0.815

Classification Report

	Precision	Recall	F1-score	Support
Fail	0.79	0.86	0.82	14792
Win	0.84	0.76	0.80	14549
Avg / Total	0.82	0.81	0.81	29341

Random Forest – Hyper Parameter Tuning

Using Python sklearn RandomizedSearchCV():

```
Parameters = {'n_estimators': [int(x) for x in np.linspace(start=10, stop=100, num=10)],
              'max_features': ['auto', 'sqrt'],
              'max_depth': [int(x) for x in np.linspace(10, 110, num = 11)],
              'min_samples_split': [2, 3, 5, 10],
              'min_samples_leaf': [1, 2, 3, 4],
              'bootstrap': [True, False]}
```

Best Hyper Parameters

{'bootstrap': False,
'max_depth': 80,
'max_features': 'sqrt',
'min_samples_leaf': 1,
'min_samples_split': 5,
'n_estimators': 60}

Tuned Random Forest Model – Red Side Win

Feature Set II – found to have slightly better results in accuracy

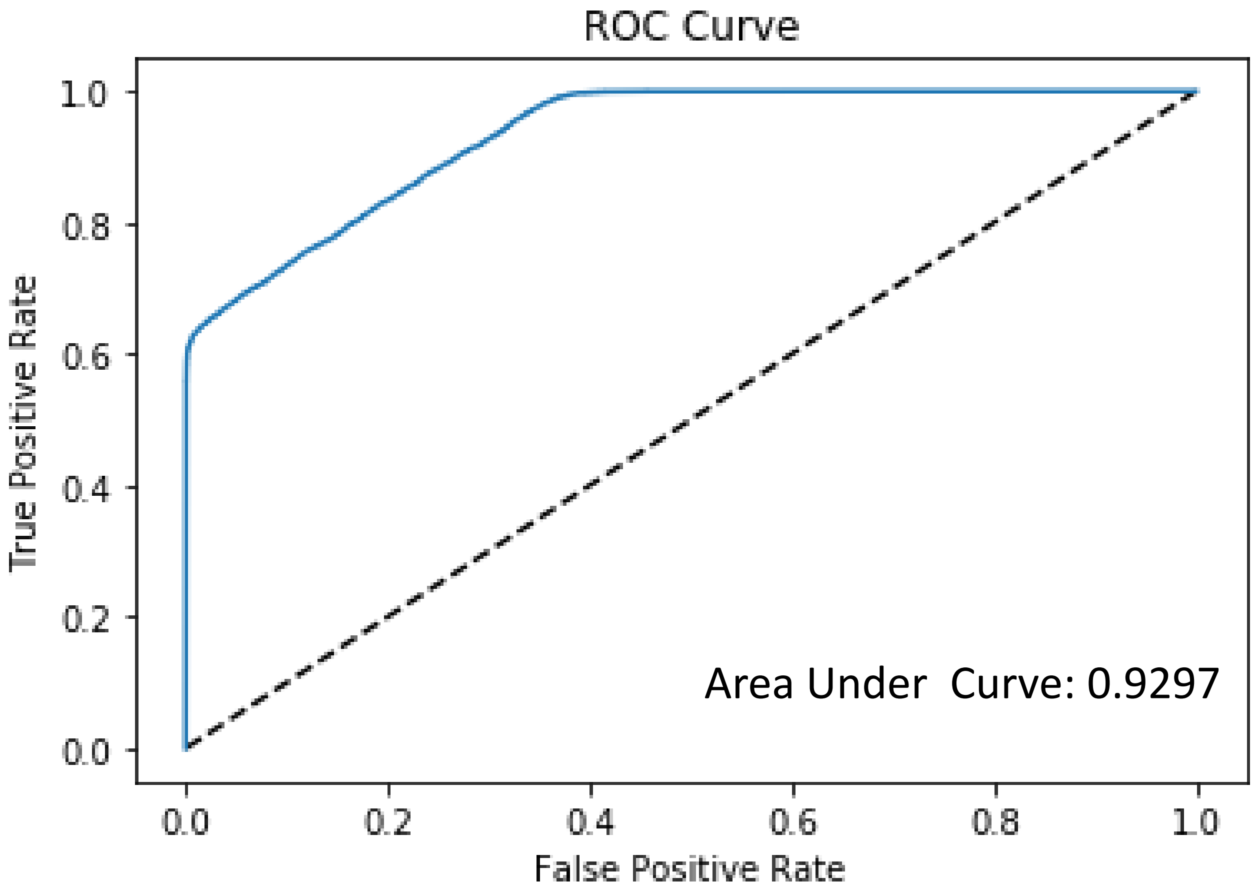
Set test size: 20%

5-Fold Cross Validation, Accuracy Score: 0.8199

Classification Report

	Precision	Recall	F1-score	Support
Fail	0.82	0.82	0.82	14336
Win	0.82	0.81	0.82	14219
Avg / Total	0.82	0.82	0.82	28555

Receiver Operator Characteristic Curve – Red Side Win



*Area under curve describes how capable the model is at distinguishing between binary classes. ('Win' or 'Fail') (0 or 1)
*The higher the score, the better the prediction.

Tuned Random Forest Model – Blue Side Win

Feature Set II

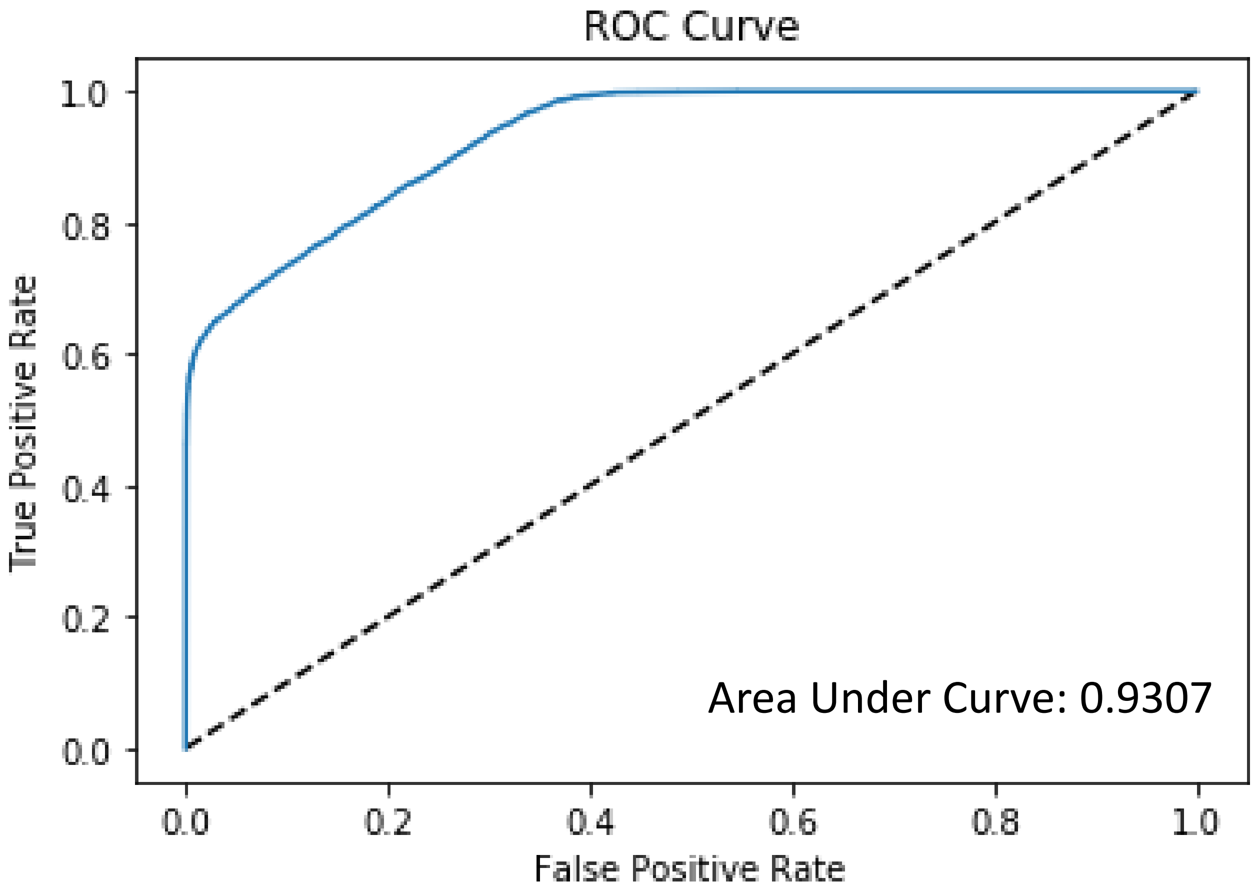
Set test size: 20%

5-Fold Cross Validation, Accuracy Score: 0.8175

Classification Report

	Precision	Recall	F1-score	Support
Fail	0.82	0.82	0.82	14219
Win	0.82	0.82	0.82	14336
Avg / Total	0.82	0.82	0.82	28555

Receiver Operator Characteristic Curve – Blue Side Win



Use Cases

- League of Legends is a free to play game. Therefore, keeping the players entertained is key in retaining a high player base. Riot games, specifically its shout casters, can use these predictions as another topic of discussion in professional tournaments. It could be a big talking point in pre-match discussion. This allows for further audience engagement.
- The machine learning predictions can be built into apps for professional players to use. Teams can input team compositions, spells, and rune choices to get better insights on creating new strategies.

Thoughts on Model Results

- Only applicable in High levels of play – too much variation in player skill and game knowledge in low level play
- A separate model/data will be needed to predict match outcomes for each tier and region, as well as professional play. Model in this project would overfit for new data not from high ranked NA games.
- Will most likely get poor results if using data from all tiers and regions. The model will most likely not generalize trends over all skill tiers.
- Current model can not account for player emotions – ‘Rage quitting’, ‘intentionally feeding’, and AFK

Improving the Model

- In the future, remove ‘One Trick’ players from data – ‘One Trick’ players play one champion only and have higher win rates than normal on the champion
- MORE DATA – API rate limit can prevent gathering large amounts of data if under a time constraint
- If making a model for professional play, then add features such as player win rates for a given champion and win rates versus specific teams

Appendix

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$\text{F1} = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

True Positive (TP): • Reality: A wolf threatened. • Shepherd said: "Wolf." • Outcome: Shepherd is a hero.	**False Positive (FP):** • Reality: No wolf threatened. • Shepherd said: "Wolf." • Outcome: Villagers are angry at shepherd for waking them up.
False Negative (FN): • Reality: A wolf threatened. • Shepherd said: "No wolf." • Outcome: The wolf ate all the sheep.	**True Negative (TN):** • Reality: No wolf threatened. • Shepherd said: "No wolf." • Outcome: Everyone is fine.

*above image https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative

Thank You!

- Justin Tsao
- Full Project: https://github.com/jltsao88/Capstone_Project_1
- Email: jltsao88@gmail.com
- https://www.linkedin.com/in/justintsao/