

# League of Legends Item Balancing Recommendations

Sebastian Alvis

25 April 2019

PuPPy: Scientific Computing Night



# Outline

- Crash Course: League of Legends
- Problem Statement
- Data
- Statistics
- Machine Learning
- Conclusions

# Outline

- Crash Course: League of Legends
- Problem Statement
- Data
- Statistics
- Machine Learning
- Conclusions



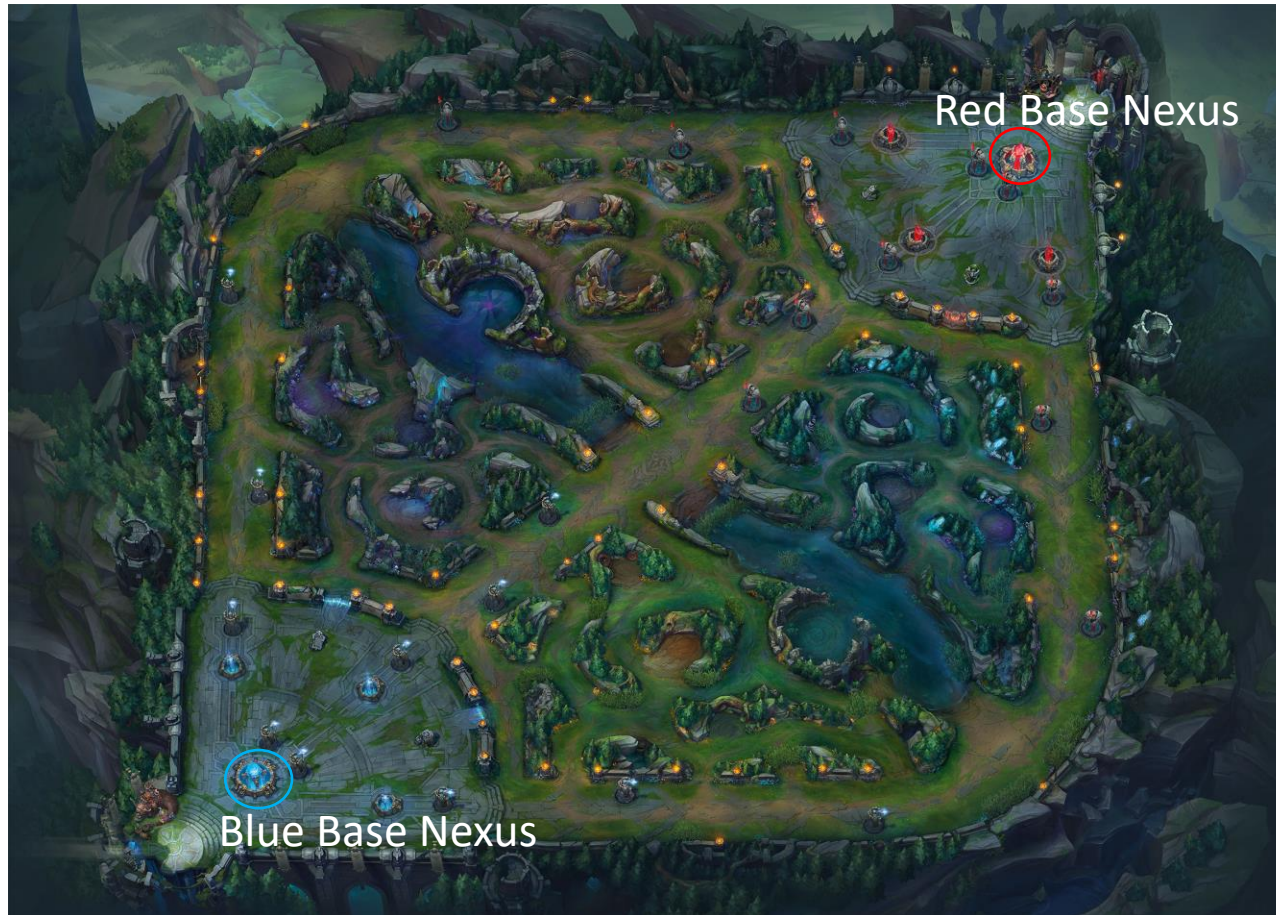
# League of Legends: The Map



- Summoner's Rift
- 5 players vs 5 players
- Both teams have bases
- 3 Lanes
- Area between lanes is the Jungle



# League of Legends: The Objective



- Objective: Destroy the Nexus

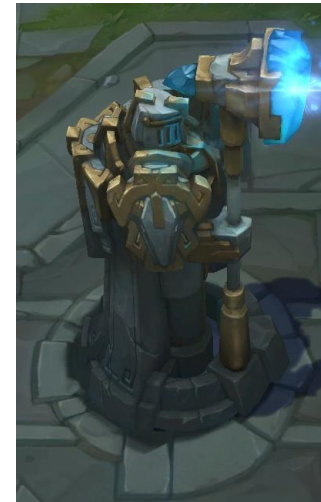




# League of Legends: Towers

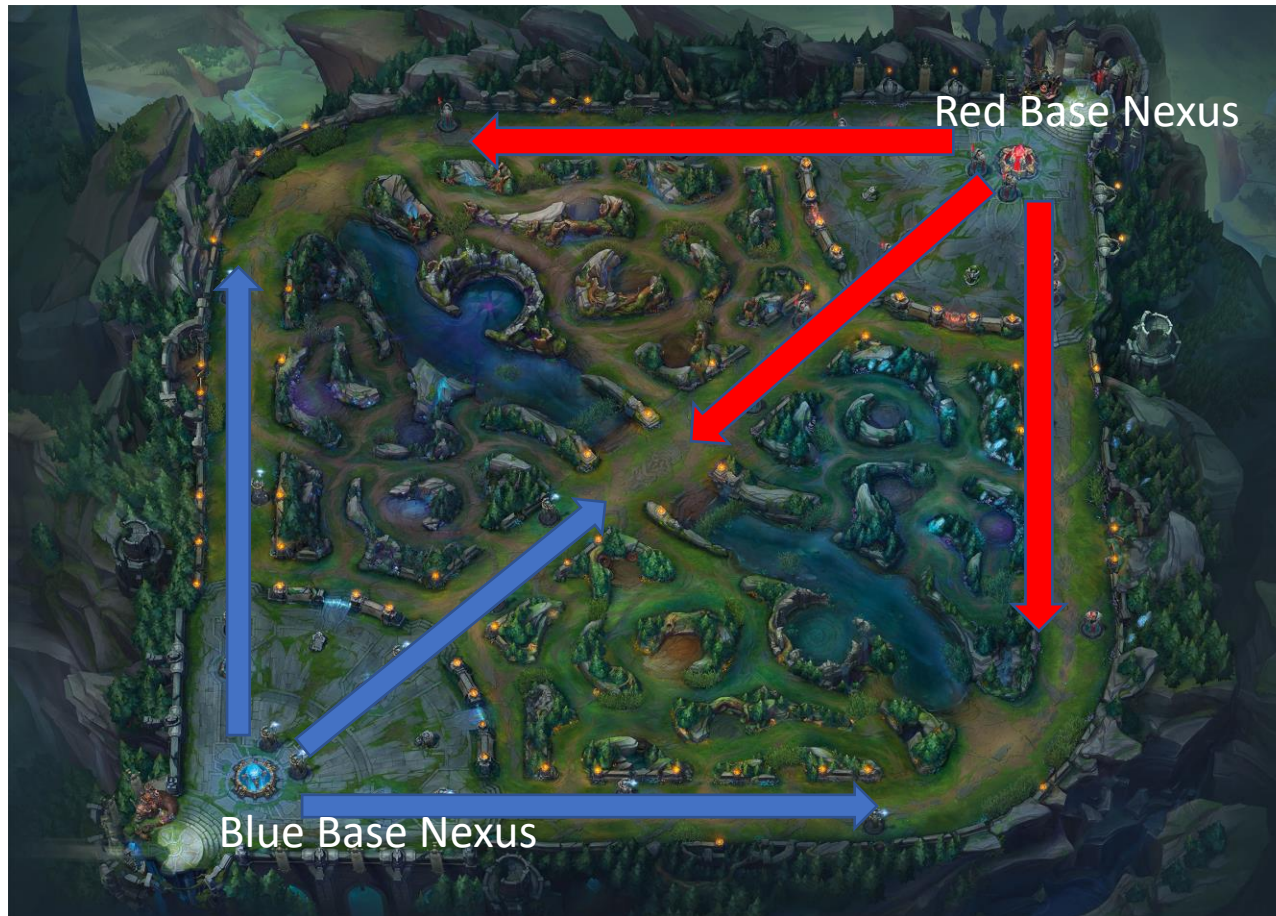


- Block the lanes
- Shoot enemy champions
- Shoot enemy minions





# League of Legends: Minions



- Spawn in every lane
- Attack enemy units in front of them
- Help get through turrets



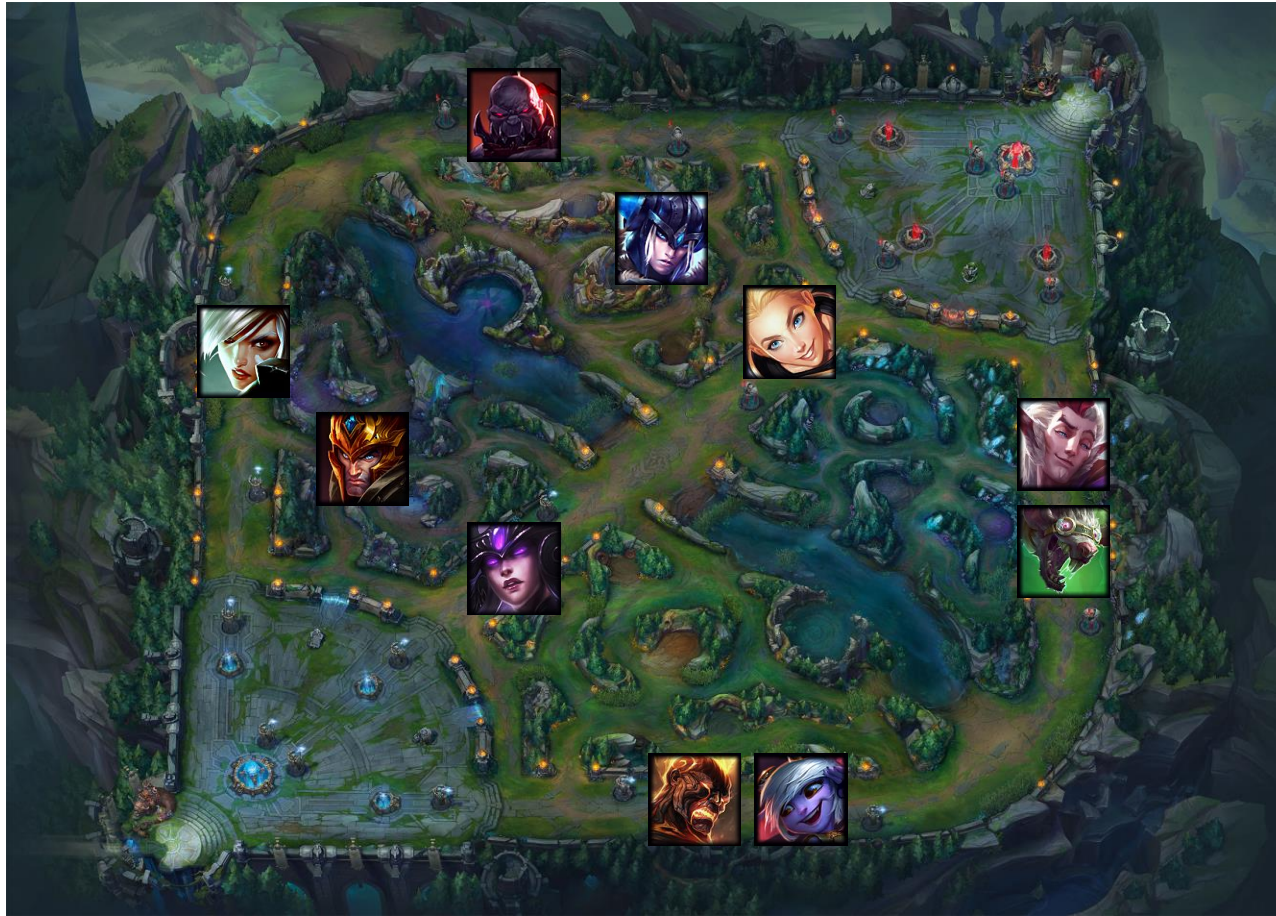
# League of Legends: Champion and Teams

Blue Team Name (Role)	Position	Red Team Name (Role)
Riven (Fighter) 	Top Lane	Sion (Tank) 
Jarvan IV (Tank) 	Jungle	Sejuani (Tank) 
Syndra (Mage) 	Mid Lane	Lux (Mage) 
Tristana (Marksman) 	AD Carry (Bot Lane)	Twitch (Marksman) 
Brand (Mage) 	Support (Bot Lane)	Rakan (Support) 

- Each champion has unique abilities
- Starting map positions for champions
  - Maximize team gold
- Various roles / archetypes for champions
  - Assassin, Fighter, Mage, Marksman, Support, Tank
  - Archetype generally determines the kinds of items you buy



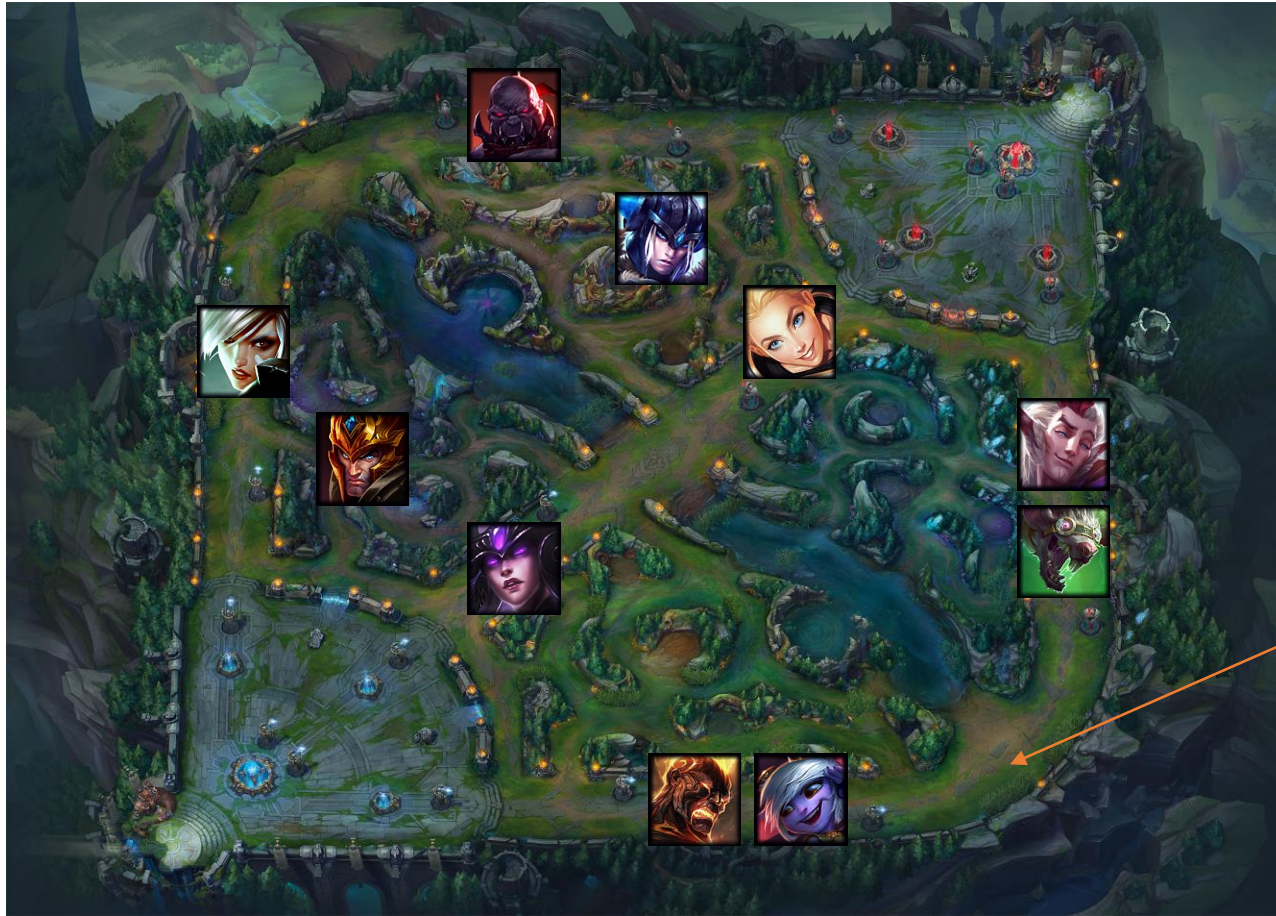
# League of Legends: Playing the Game



- Objective: Destroy the Nexus
- Make your champion stronger
  - Level up
  - Buy items
- Fighting various enemies provides gold and experience



# League of Legends: Playing the Game

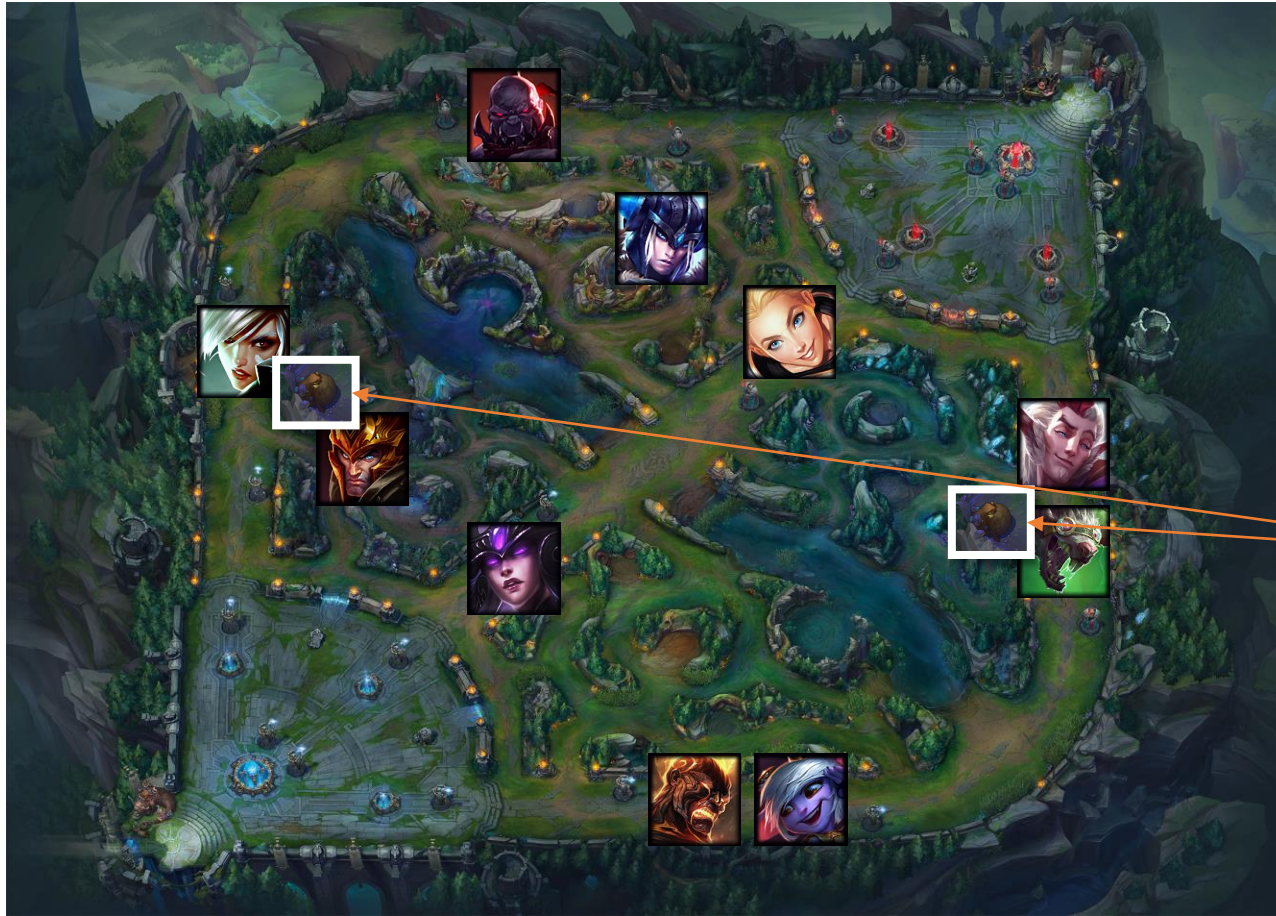


- Objective: Destroy the Nexus
- Fighting various enemies provides gold and experience
  - Enemy minions





# League of Legends: Playing the Game

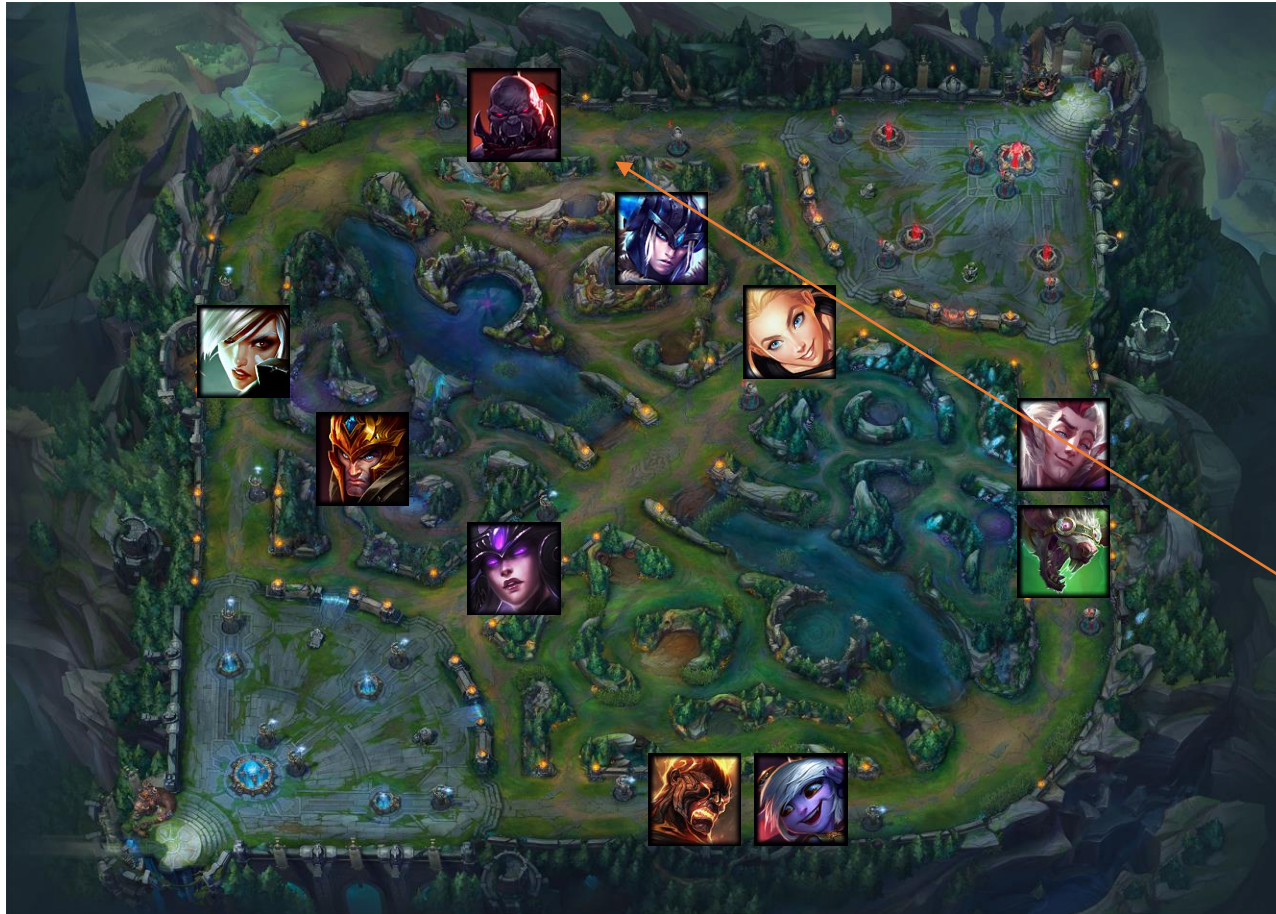


- Objective: Destroy the Nexus
- Fighting various enemies provides gold and experience
  - Enemy minions
  - Jungle monsters





# League of Legends: Playing the Game

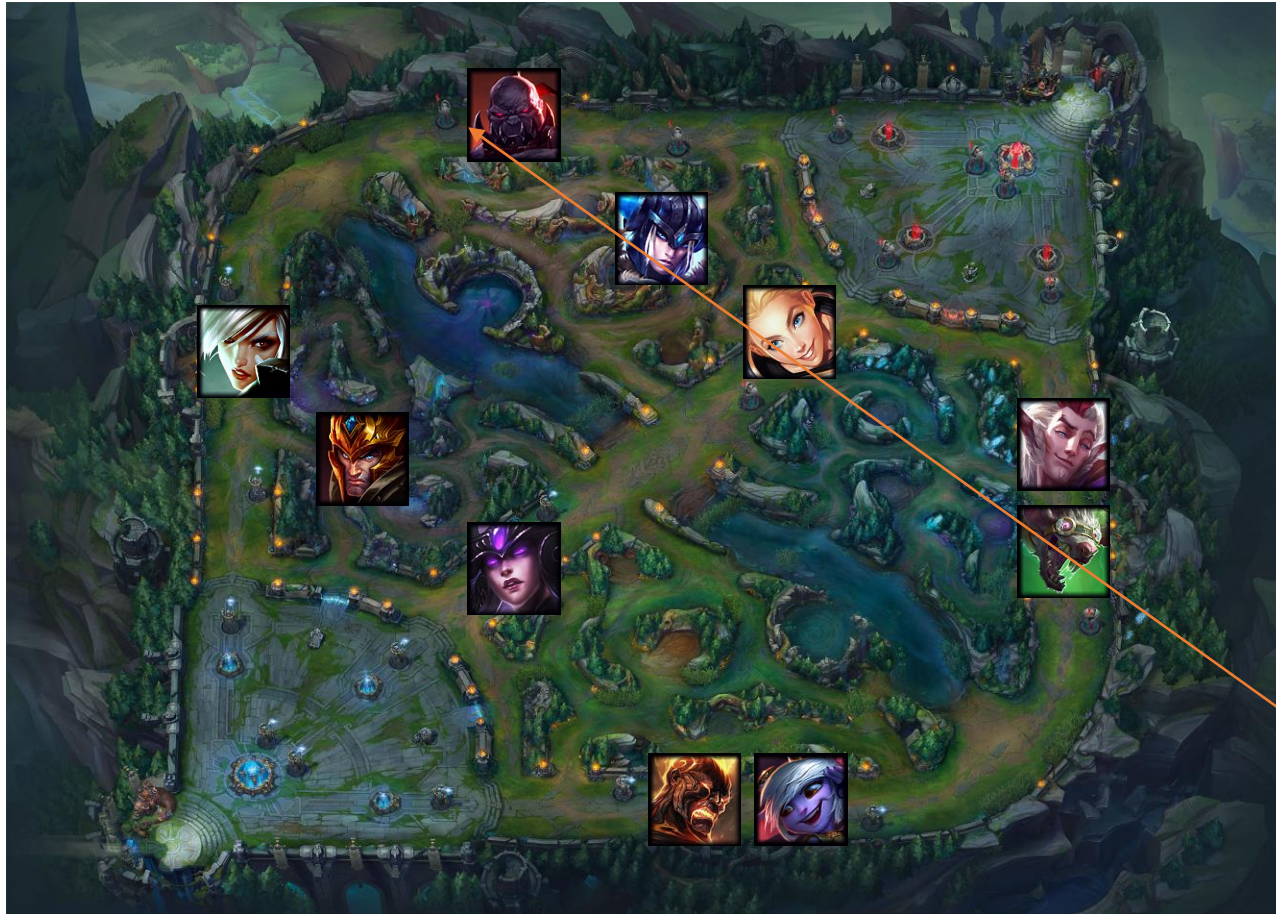


- Objective: Destroy the Nexus
- Fighting various enemies provides gold and experience
  - Enemy minions
  - Jungle monsters
  - Enemy champions





# League of Legends: Playing the Game

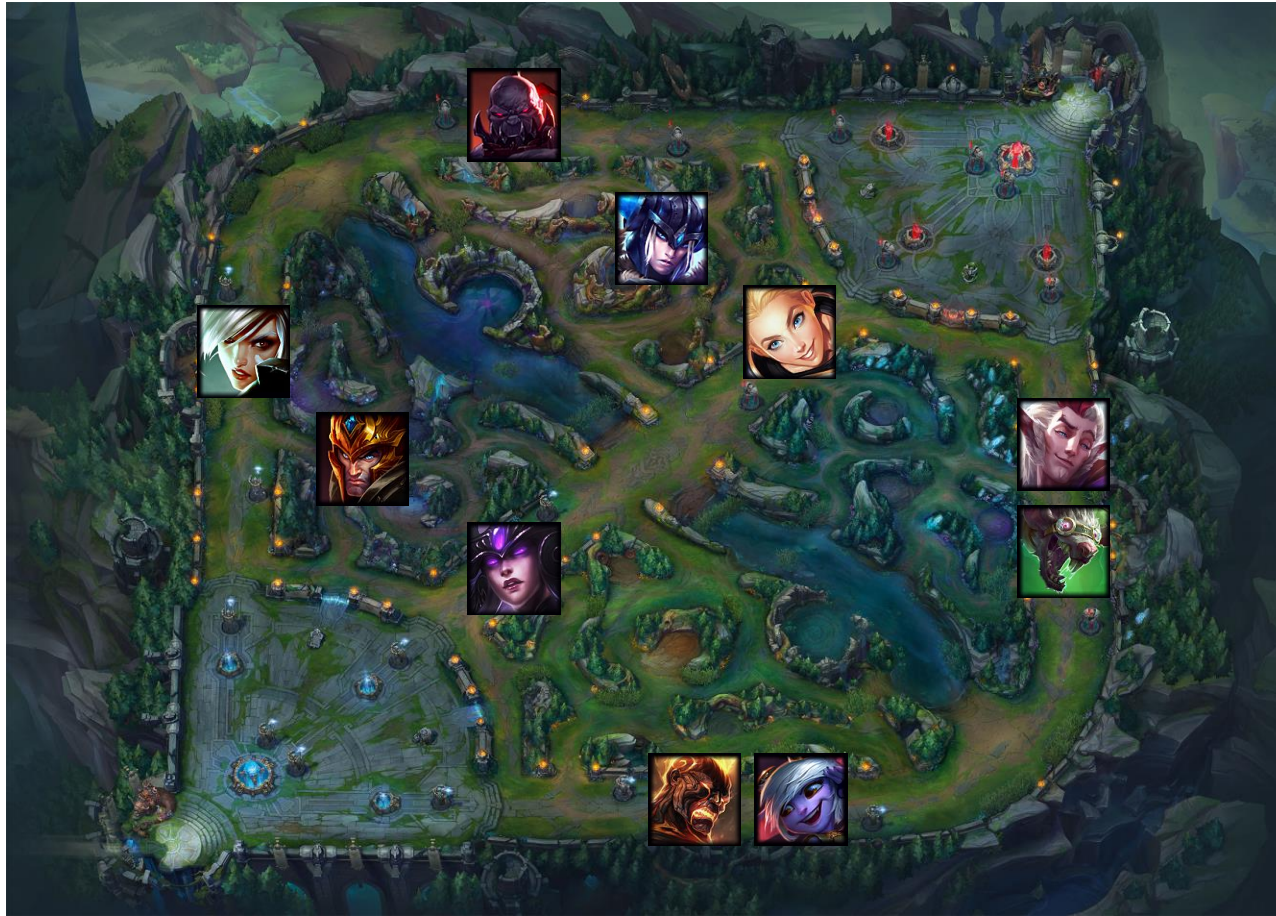


- Objective: Destroy the Nexus
- Fighting various enemies provides gold and experience
  - Enemy minions
  - Jungle monsters
  - Enemy champions
  - Towers

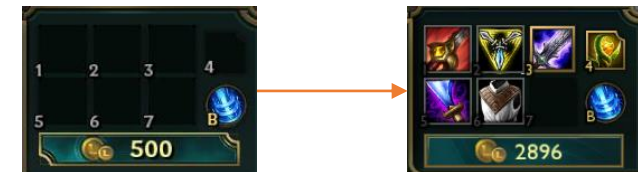




# League of Legends: Playing the Game



- Objective: Destroy the Nexus
- Fighting various enemies provides gold and experience
- Leveling up improves abilities
- Items upgrade various attributes of the champion





# Outline

- Crash Course: League of Legends
- **Problem Statement**
- Data
- Statistics
- Machine Learning
- Conclusions

# Why should you change the game at all?

- League of Legends currently has
  - 143 champions
  - 200+ items
- Diversity is it's strong suit
  - Nobody wants to see the same champions and items every game
  - Game balance causes different champions and items to show up
  - More interesting gameplay

# Problem Statement

- How do you know when you should change an item?



# Outline

- Crash Course: League of Legends
- Problem Statement
- **Data**
- Statistics
- Machine Learning
- Conclusions

# “Static” Data

- Acquired through the Riot Games Static API
  - Items
  - Champions
  - JSON files
  - It is consistent data
  - Static
- Patches can tweak champions / items
  - “Static”



# Match Data



- Taken from the Ranked Queue
  - People take the game more seriously
- Only higher-level Leagues: Diamond and Platinum
  - People have more game knowledge and experience
- 1000 random players
- 1000 random matches
- Match data
  - Game state at the end of the game

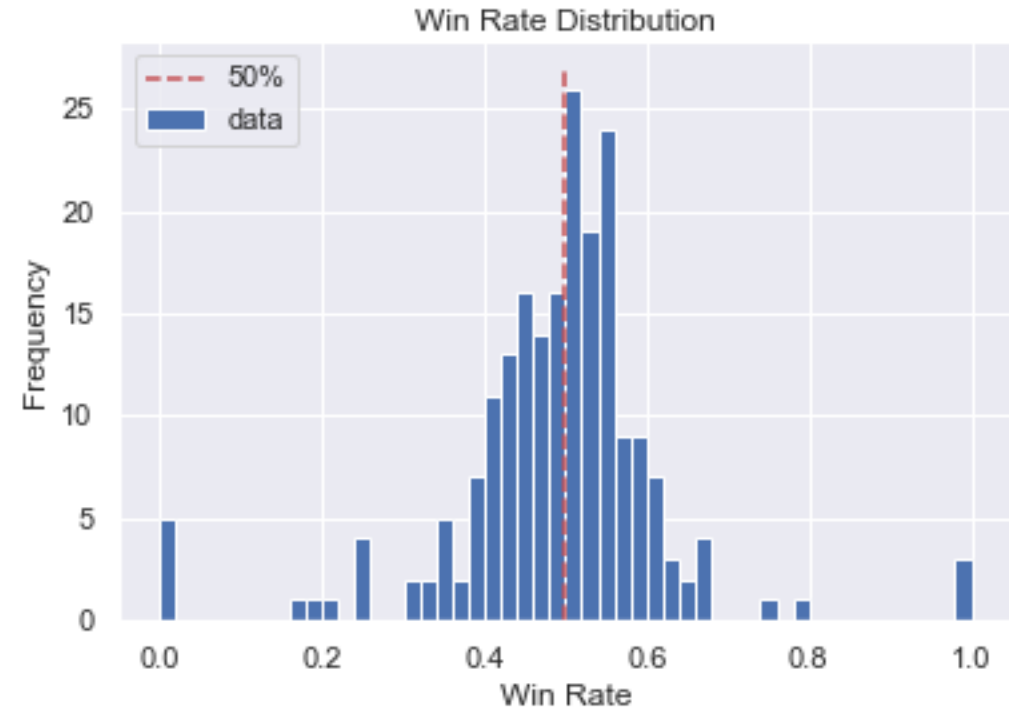


# Dataframes

- Items
- Champions
- Matches
- Item Statistics
  - Win Rate
  - Pick Rate

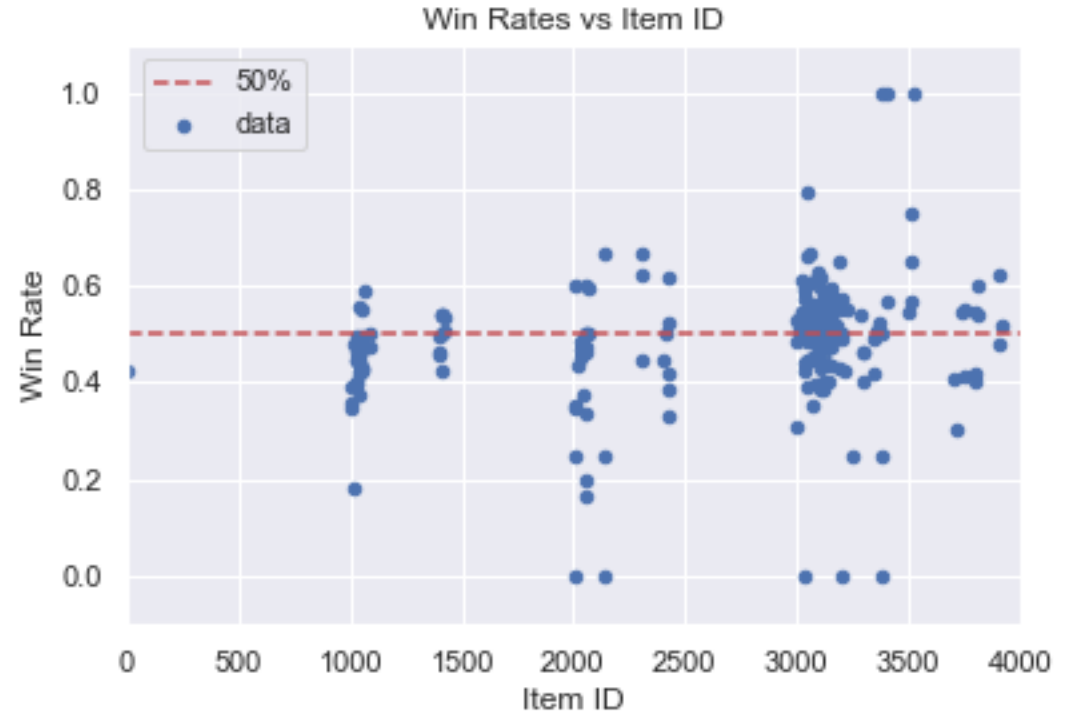
# Item Stats: Win Rate

- $\frac{\text{\# of occurrences with a win}}{\text{\# of occurrences}}$
- Bernoulli variable
- Easy Interpretation
- Avg: 0.50



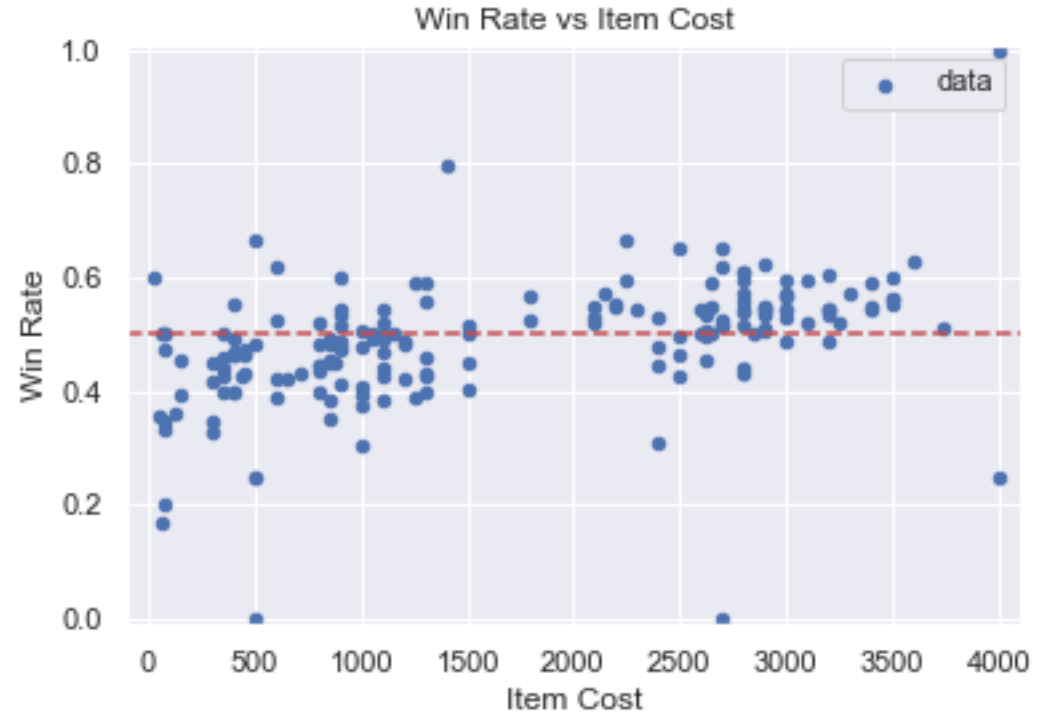
# Item Stats: Win Rate

- $\frac{\text{\# of occurrences with a win}}{\text{\# of occurrences}}$
- Bernoulli variable
- Easy Interpretation
- Avg: 0.50



# Item Stats: Win Rate vs Item Cost

- This shows that finishing an item is good for your chances of winning
- The final analysis will use mostly the right lobe
- Focusing on full items and which ones are too strong / weak





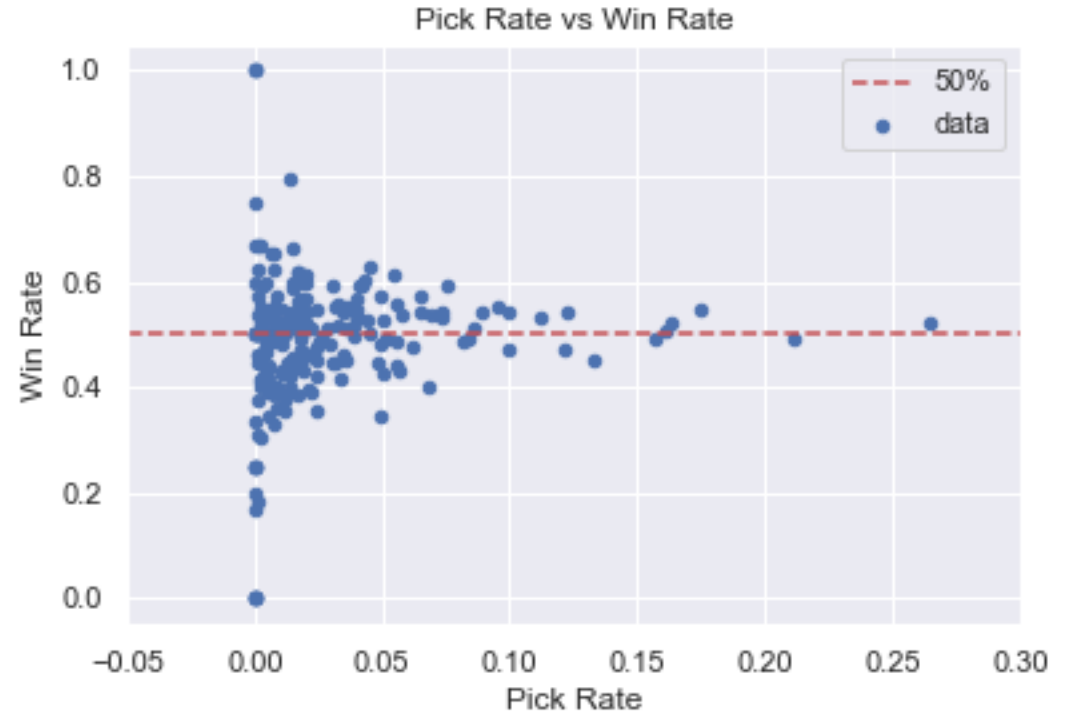
# Item Stats: Pick Rate

- How often an item is bought
- A metric of popularity
  - Popularity often indicates what is strong
  - This helps us when win rate can't
- Also pretty easy to understand



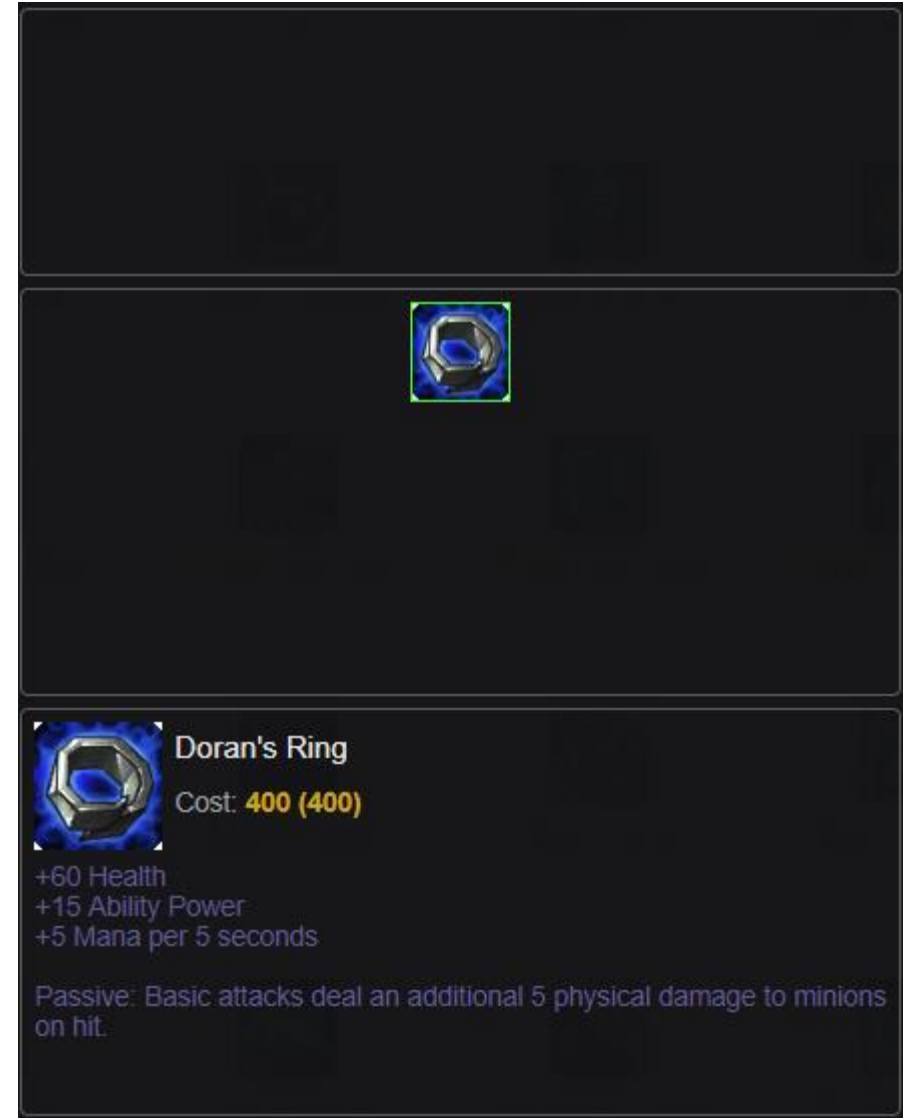
# Pick Rate vs Win Rate

- Higher pick rate
- Closer to the mean
- This drags the win rate to 50%




# Final Item Dataset

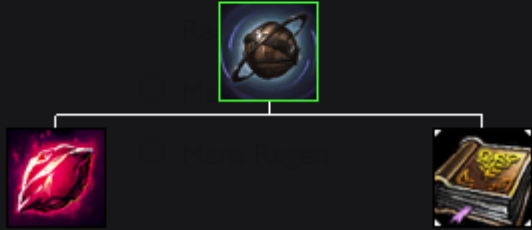
- There are 200+ items in League
- I don't want to look at all of them
- Remove
  - Starting items




# Final Item Dataset

- There are 200+ items in League
- I don't want to look at all of them
- Remove
  - Starting items
  - Incomplete items





```
graph TD; A[Oblivion Orb] --- B[Item 1]; A --- C[Item 2]
```




**Oblivion Orb**  
Cost: **1600 (765)**

+20 Ability Power  
+200 Health

UNIQUE Passive - Touch of Death: +15 [Magic Penetration](#)

# Final Item Dataset

- There are 200+ items in League
- I don't want to look at all of them
- Remove
  - Starting items
  - Incomplete items
  - Items with fewer than 10 occurrences



```
graph TD; A[Infernal Mask] --> B[Infernal Mask]; B --> C[Archangel's Staff]; B --> D[Seraph's Embrace]; C --> E[Archangel's Staff]; C --> F[Archangel's Staff]; D --> G[Seraph's Embrace]; D --> H[Seraph's Embrace];
```

**Infernal Mask**  
Cost: 3000 (0)

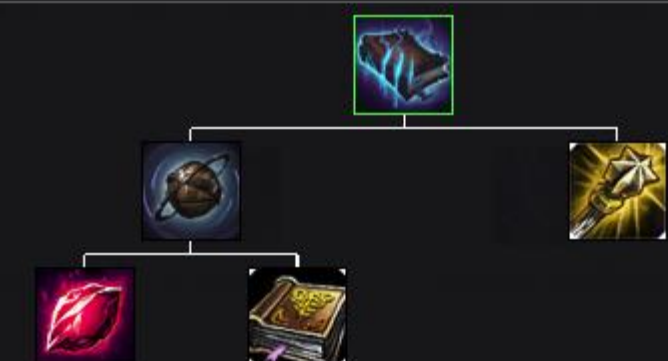
+750 Health  
+300 Mana  
**+100 Magic Resist**  
+10% Cooldown Reduction


UNIQUE Passive - Eternity: 15% of damage taken from champions is gained as Mana. Spending Mana restores 20% of the cost as Health, up to 25 per spell cast.



# Final Item Dataset

- There are 200+ items in League
- I don't want to look at all of them
- Remove
  - Starting items
  - Incomplete items
  - Items with fewer than 10 occurrences
- Resultant table has 80 items



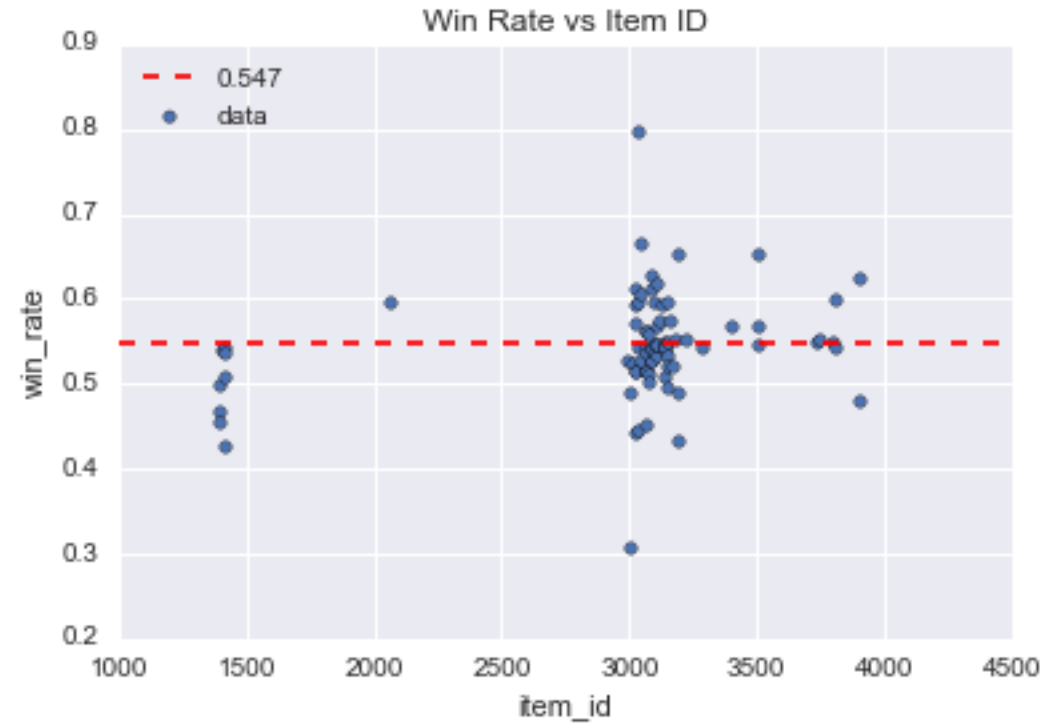
 **Morellonomicon**  
Cost: **3000 (550)**

+70 Ability Power  
+300 Health

UNIQUE Passive - Touch of Death: +15 [Magic Penetration](#)  
UNIQUE Passive - Cursed Strike: Magic damage dealt to champions inflicts them with [Grievous Wounds](#) for 3 seconds.

# Final Item Win Rates

- $\frac{\text{\# of occurrences with a win}}{\text{\# of occurrences}}$
- Bernoulli variable
- Easy Interpretation
- Weighted Avg: 0.547



# Outline

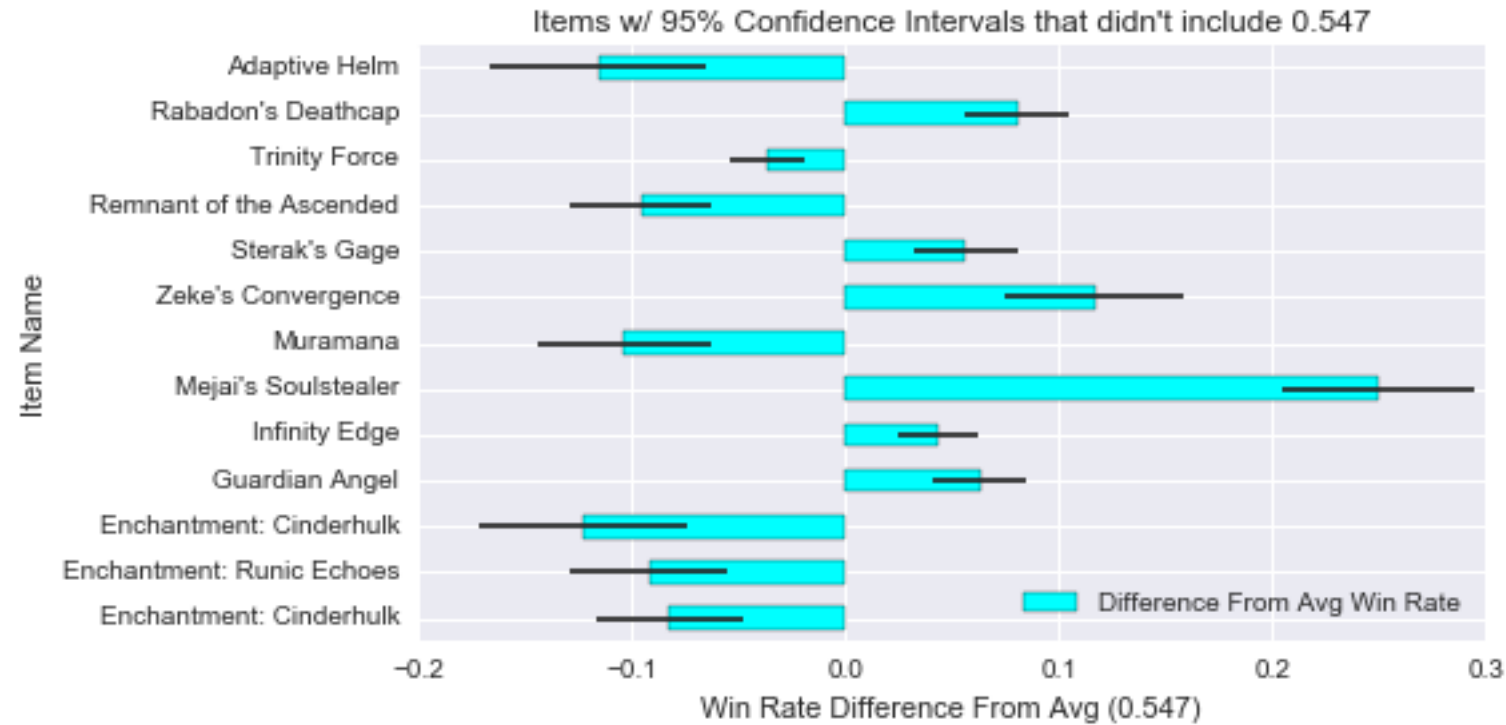
- Crash Course: League of Legends
- Problem Statement
- Data
- **Statistics**
- Machine Learning
- Conclusions



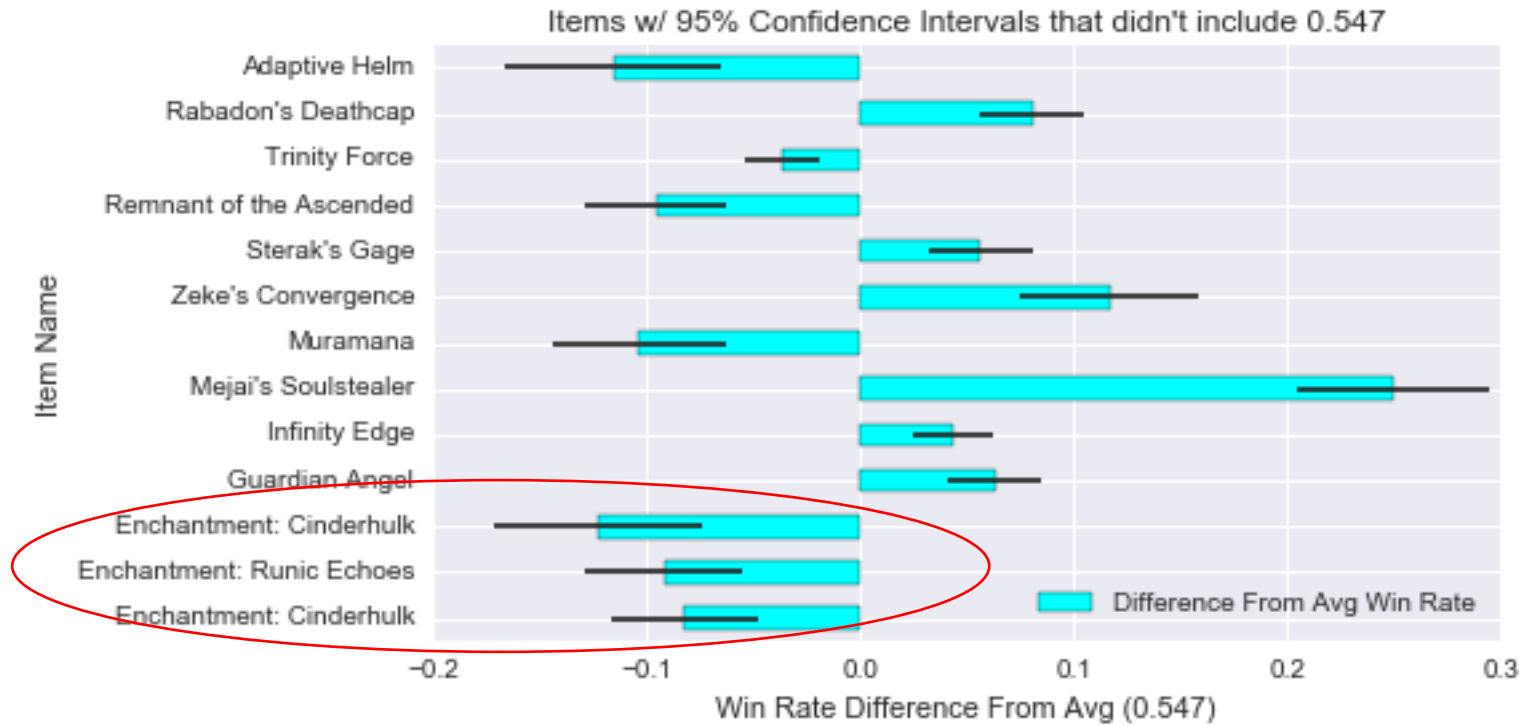
# Statistics: Abnormal Win Rates

- Win rate is Bernoulli
  - Easy to estimate a sample std
- Avg: 0.547
- 95% Confidence Interval for each items' win rate
- Abnormal Items:
  - Items where the mean isn't inside the 95% CI

# Statistics: Abnormal Win Rates






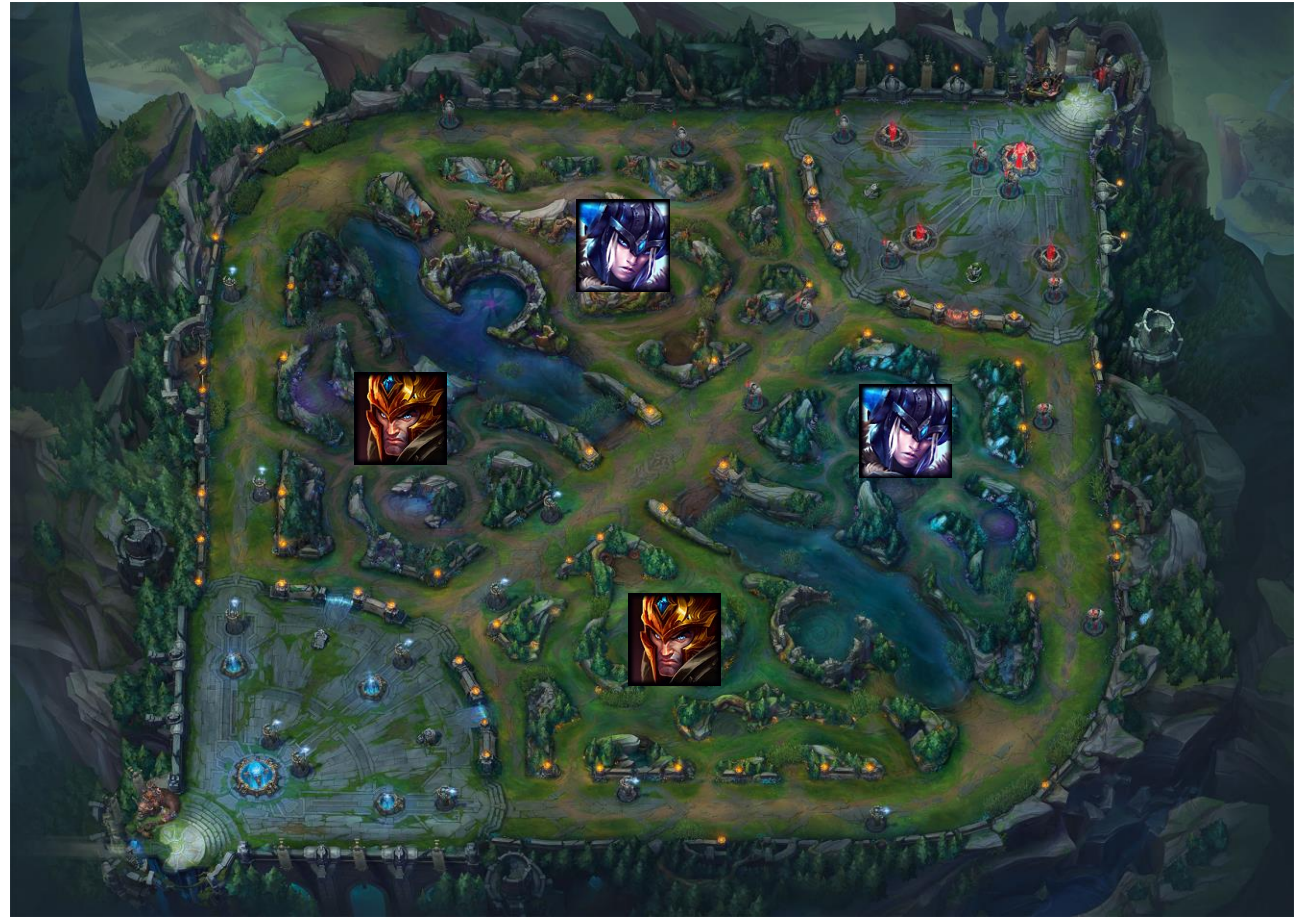
# Statistics: Abnormal Win Rates





# Statistics: Junglers

Blue Team Name (Role)	Position	Red Team Name (Role)
Riven (Fighter) 	Top Lane	Sion (Tank) 
Jarvan IV (Tank) 	Jungle	Sejuani (Tank) 
Syndra (Mage) 	Mid Lane	Lux (Mage) 
Tristana (Marksman) 	AD Carry (Bot Lane)	Twitch (Marksman) 
Brand (Mage) 	Support (Bot Lane)	Rakan (Support) 



- 'Enchantments' belong to the Jungle Item Group

# Statistics: Jungle Items



The diagram shows the item tree for Cinderhulk. At the top is the Cinderhulk icon. It branches into two items: a Flame icon and a Dagger icon. The Flame icon branches into a Flame icon and a Flame icon. The Dagger icon branches into a Dagger icon and a Dagger icon.

 **Enchantment: Cinderhulk**  
Cost: **2500 (600)**

+300 Health  
+15% Bonus Health

UNIQUE Passive - Immolate: Deals 11 (+1 per champion level) magic damage a second to nearby enemies while in combat. Deals 200% bonus damage to minions and monsters.

Limited to 1 Gold Income or Jungle item.



The diagram shows the item tree for Cinderhulk. At the top is the Cinderhulk icon. It branches into two items: a Flame icon and a Dagger icon. The Flame icon branches into a Flame icon and a Flame icon. The Dagger icon branches into a Dagger icon and a Dagger icon.

 **Enchantment: Cinderhulk**  
Cost: **2500 (600)**

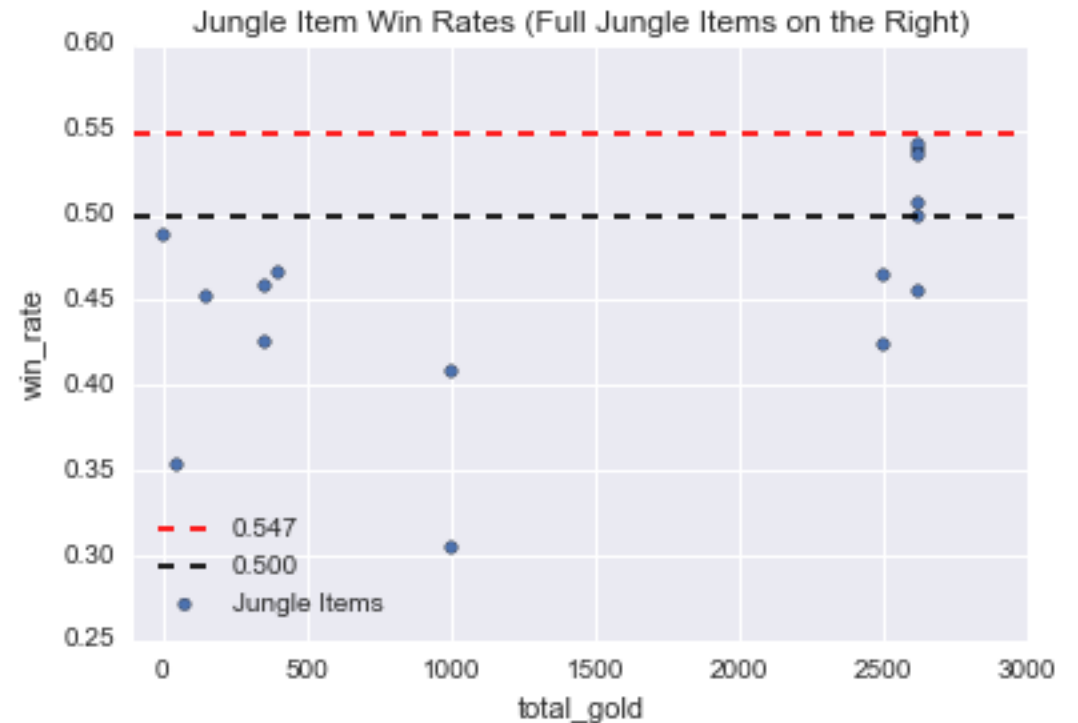
+300 Health  
+15% Bonus Health

UNIQUE Passive - Immolate: Deals 11 (+1 per champion level) magic damage a second to nearby enemies while in combat. Deals 200% bonus damage to minions and monsters.

Limited to 1 Gold Income or Jungle item.

# Statistics: Jungle Items

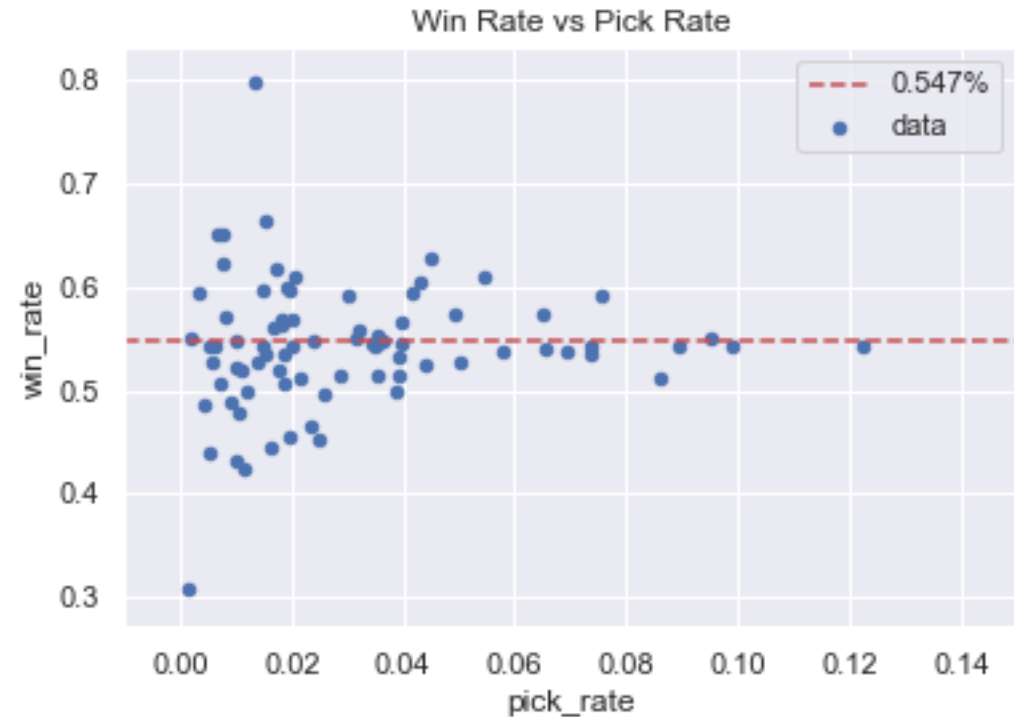
- Win Rate is not centered around 54.7% like the rest of the items
- Win Rate should be centered around 50%





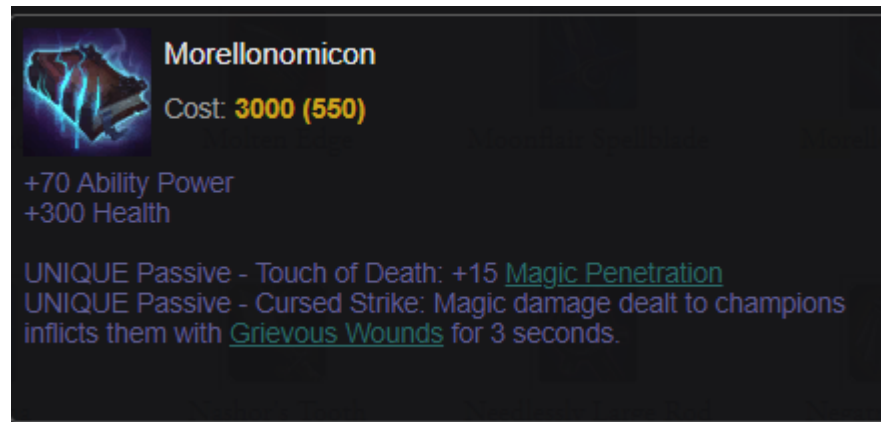
# Statistics: Win Rate Noise





- Item picked once on each side
- Win rate is 0.50
- Not informative
- Win rate drifts towards 0.50
  - “Masking”



# Statistics: Win Rate Noise

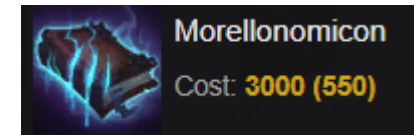
- Item picked once on each side
- Win rate is 0.50
- Not informative
- Win rate drifts towards 0.50
  - “Masking”



Blue Team Name (Role)	Position	Red Team Name (Role)
Riven (Fighter) 	Top Lane	Sion (Tank) 
Jarvan IV (Tank) 	Jungle	Sejuani (Tank) 
Syndra (Mage) 	Mid Lane	Lux (Mage) 
Tristana (Marksman) 	AD Carry (Bot Lane)	Twitch (Marksman) 
Brand (Mage) 	Support (Bot Lane)	Rakan (Support) 

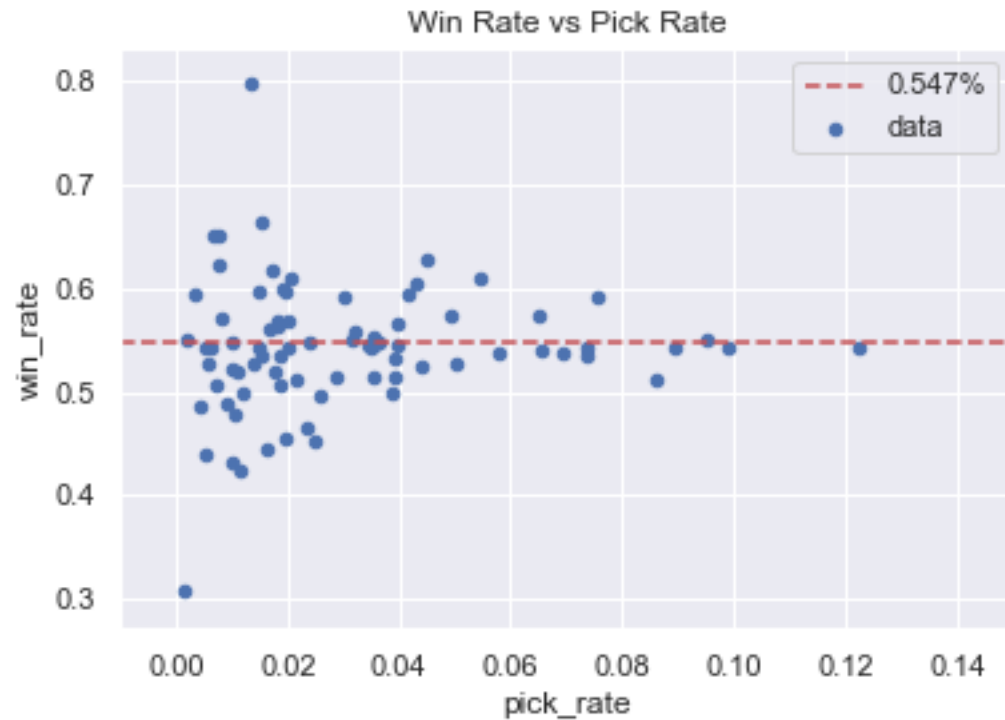
# Statistics: Unmask Win Rates

- Loop for each item
  - Find how many more one team built than the other
  - Track the number of wins and total games from that difference
- Ex:
  - Team 1 builds 3 Morellonomicons and Loses
  - Team 2 builds 1 Morellonomicon and Wins
  - The loop only counts 2 games for Morellonomicon
    - Both losses

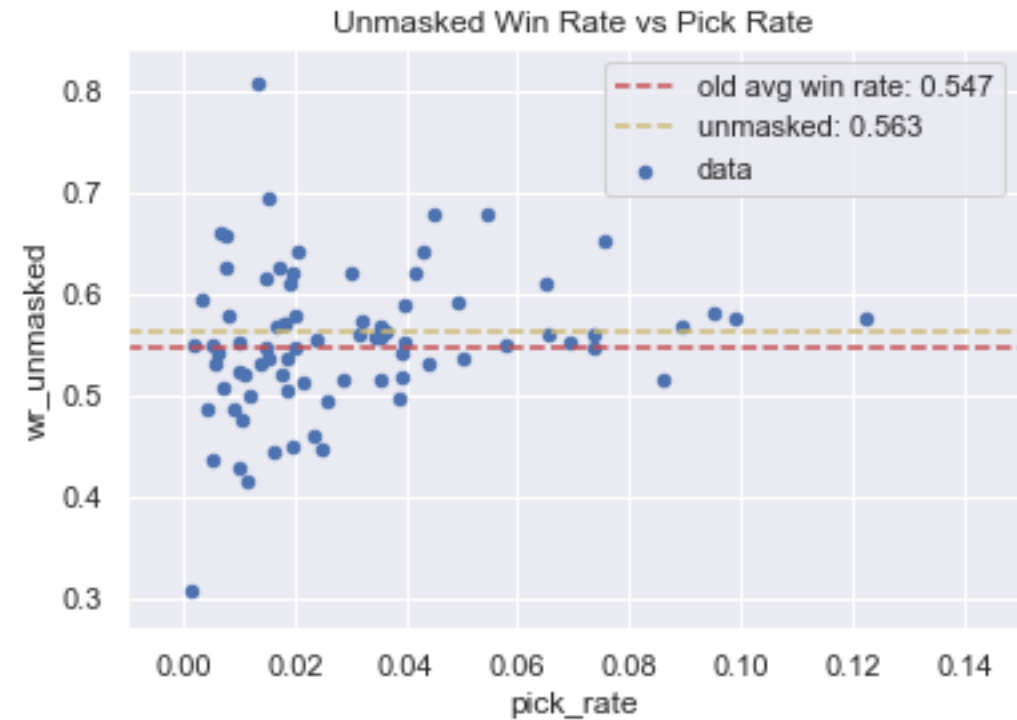


# Statistics: Unmasked Win Rates

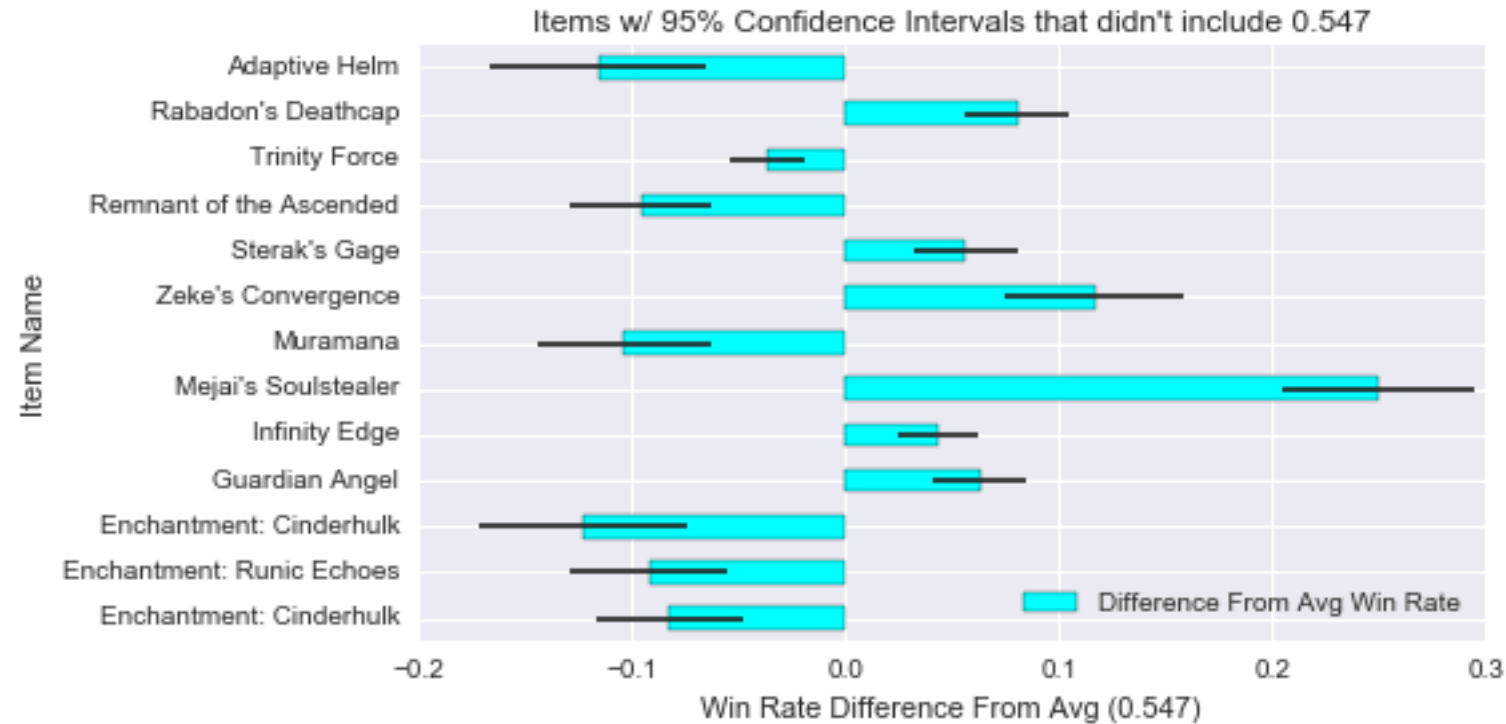
- Before Unmasking



- After Unmasking

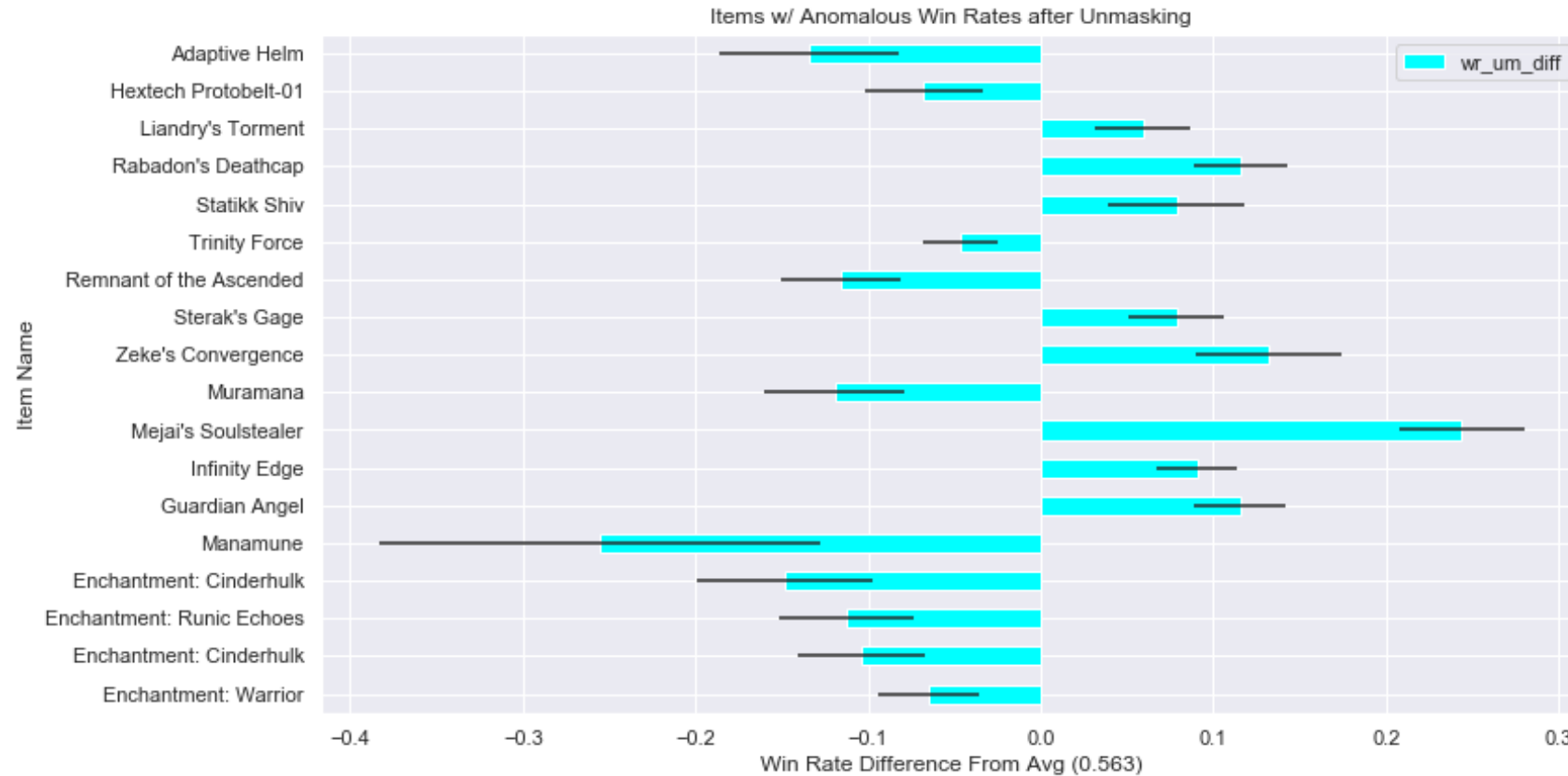


# Statistics: Abnormal Win Rates





# Statistics: Abnormal Win Rates 2.0



- Same analysis as before, but with new win rates and counts

# Statistics: Results

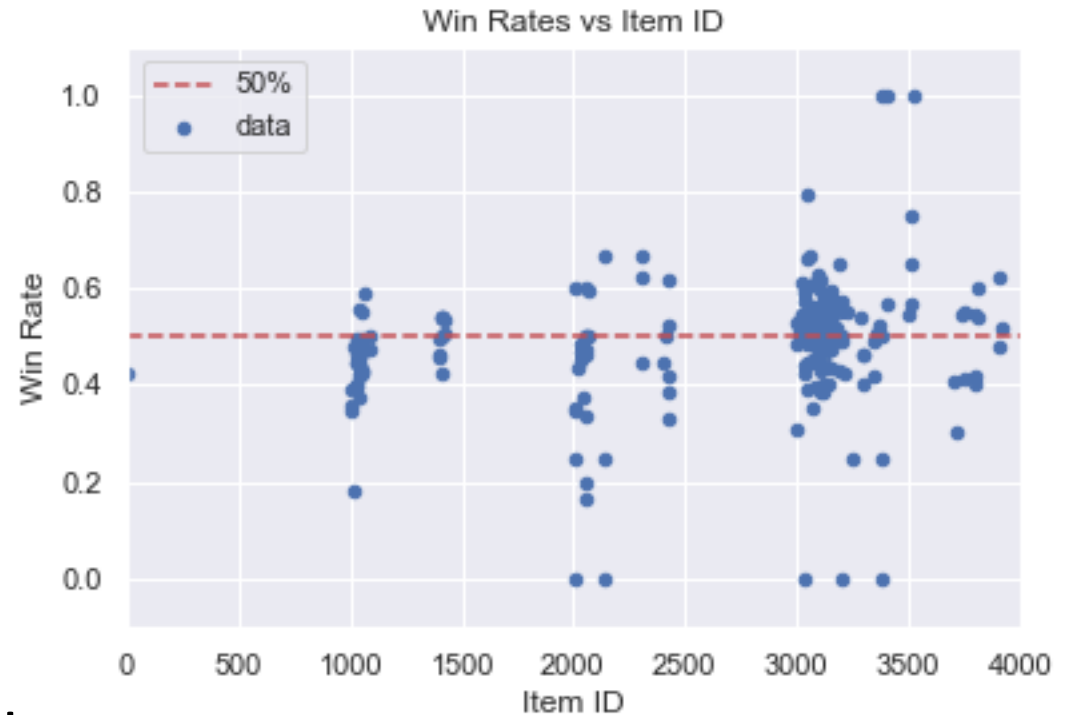
- Two sets of items with anomalous win rates
  - Items that seem too powerful
  - Items that seem too weak
- 18 items with anomalous win rates out of 80
  - 5 items were found to be anomalous only after unmasking
  - 4 items are jungle items, so maybe I let them slide

# Outline

- Crash Course: League of Legends
- Problem Statement
- Data
- Statistics
- **Machine Learning**
- Conclusions

# Machine Learning: Considerations

- Item ID is a bad variable to use
  - Numeric, but
  - The number is meaningless
  - Make it categorical?
- Considered using
  - Item Costs
  - Different slices of an inventory
  - Item statistics (win rate, pick rate)
- The potential for correlations is high
  - Danger



# Machine Learning: Random Forests

- Basic Random Forest on win/loss and item ID

- Accuracy of 0.53
- Essentially guessing the mean
- Garbage

	win	item_id
0	False	3020
1	False	3117
2	False	1416
3	False	3020
4	False	3078

- GridSearch with 5-fold CV

- Accuracy of 0.53 as well

- This model has so little data

- No wonder it can't do anything



# Machine Learning: Considerations

- A single inventory isn't a good predictor
  - I'm not using timing data right now
  - Needs a sort of normalization
- Solution: Team-Wide Item Differences
  - See how many of each item an entire team has
  - Compare that to the other team
  - Similar to my process for unmasking
- Fixes the masking problem
- Takes into account that LoL is a team game

# Machine Learning: Feature Generation

- Row for each game
- Column for each item ID (including 0, Empty Item Slot)
- Value is the # of that item team 1 has – the # of that item team 2 has
  - Positive values indicate team 1 has more of the item
  - Negative values indicate team 2 has more of the item
  - 0 indicates that same number of them item on both teams

	1402	1412	3027	3068	3069	3071	3092	3100	3153	3157	3285	3504	3800	3814	0
8	1.0	-1.0	-1.0	-1.0	1.0	-1.0	-1.0	-1.0	-1.0	1.0	1.0	-1.0	-1.0	-1.0	3.0

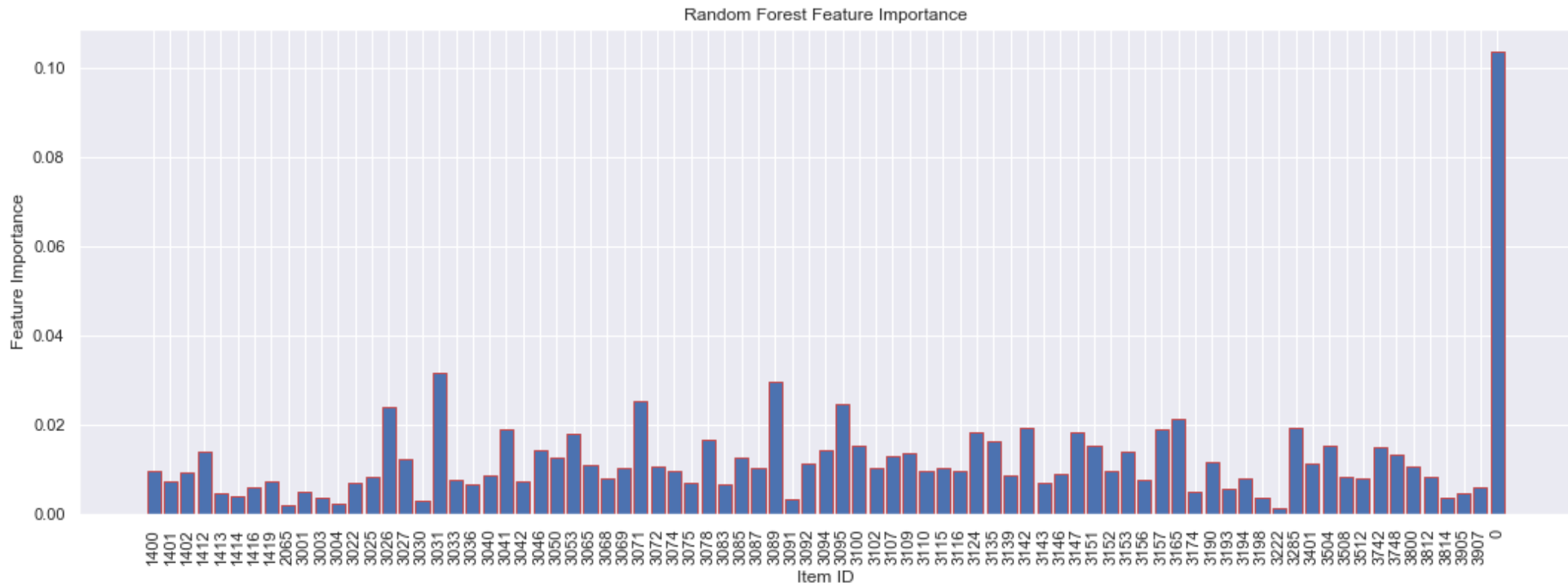
- Predictor is the match result: 0 or 1

# Machine Learning: Random Forests 2.0

- Basic Random Forest Classifier
  - Features: 'auto'
  - n\_estimators: 10
  - Accuracy: 0.69
- Best RF from Grid Search with 5 – fold CV
  - Features: 'auto' (really it was 81)
  - n\_estimators: 50
  - Accuracy: 0.73

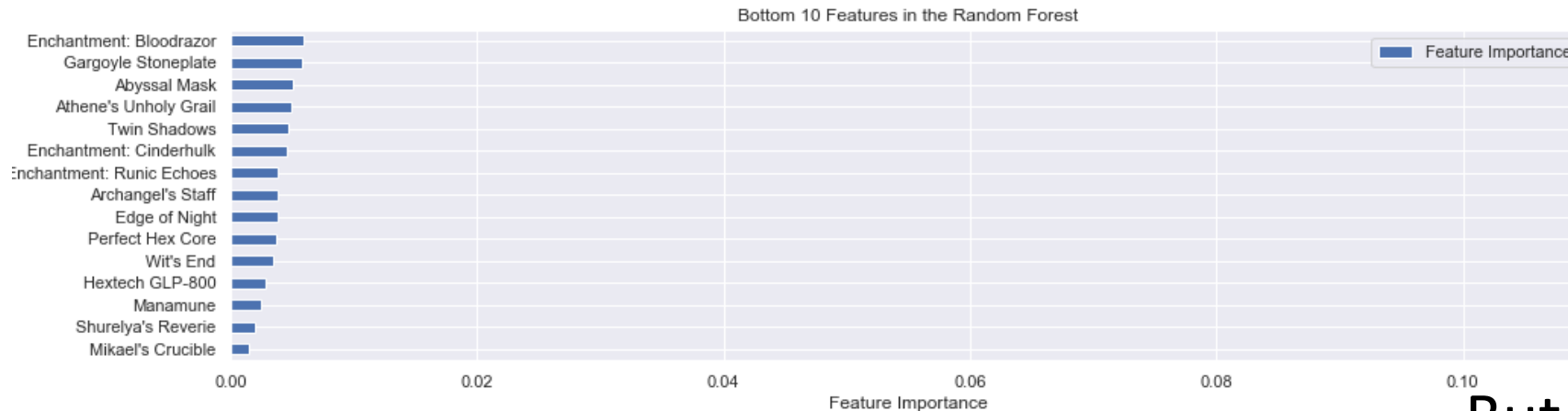
# Machine Learning: Random Forests 2.0

- Random Forest chose to have each column be a feature
  - Each item is a feature!
  - Easy to interpret? Kind of

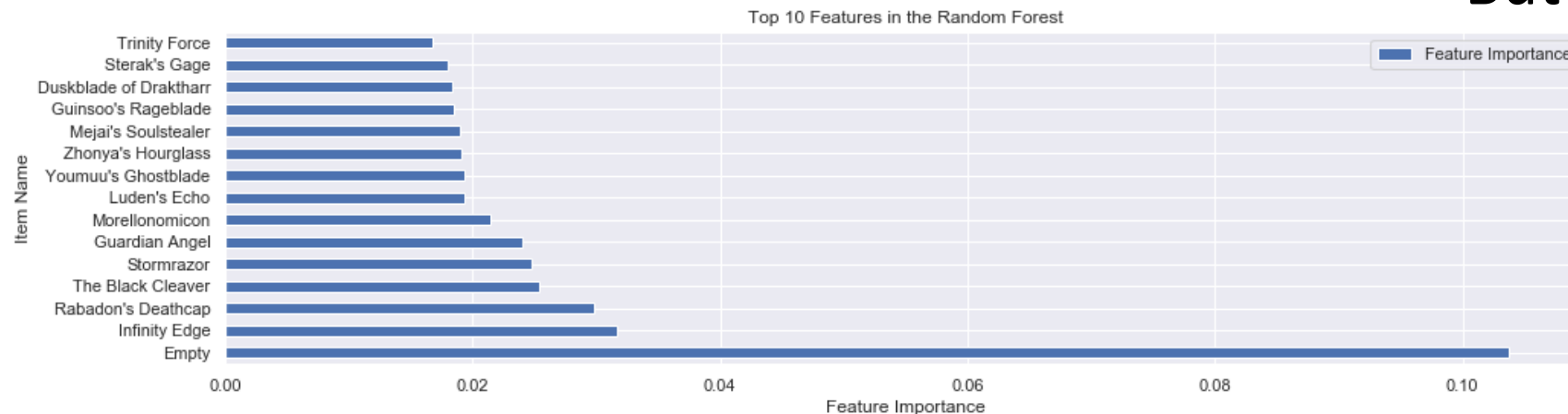


# Machine Learning: Random Forests 2.0

- I can find the most and least important features



But what do they mean?





# Machine Learning: Logistic Regression

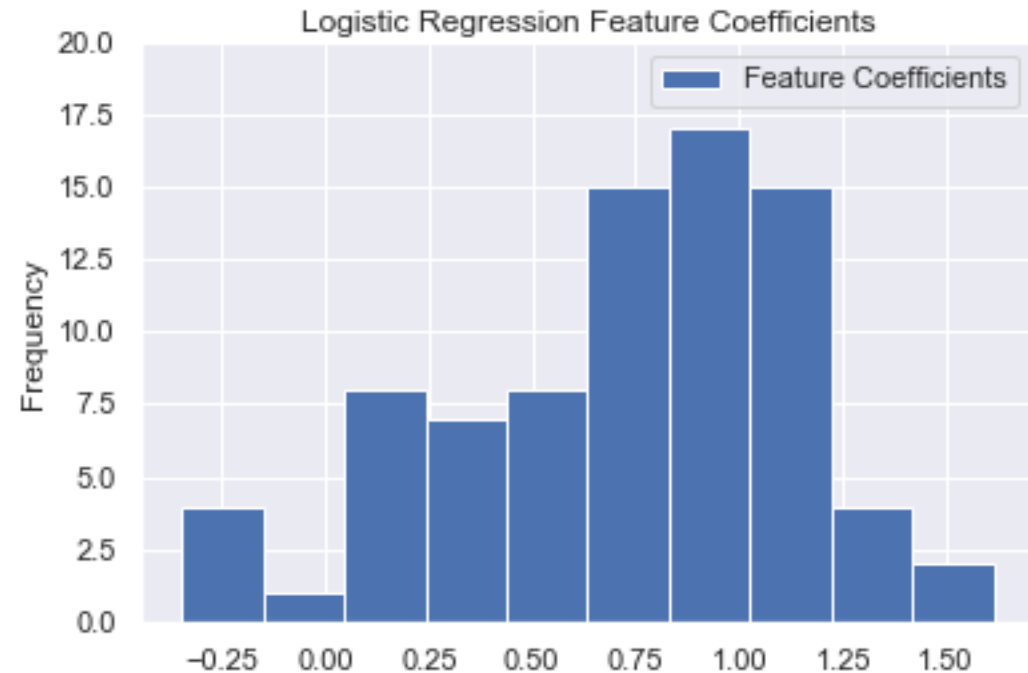
- Use the same data
- Advantages over Random Forests
  - Easier to interpret
  - Naturally focuses on binary predictions
  - Should be more accurate

# Machine Learning: Logistic Regression

- Basic Logistic Regression
  - L2 Penalty
  - Penalty Normalization C: 1.0
  - Training Accuracy: 0.92
  - Test Accuracy: 0.89
- Huge Improvements over Random Forests
- Grid Search Returned the same parameters as the default LR

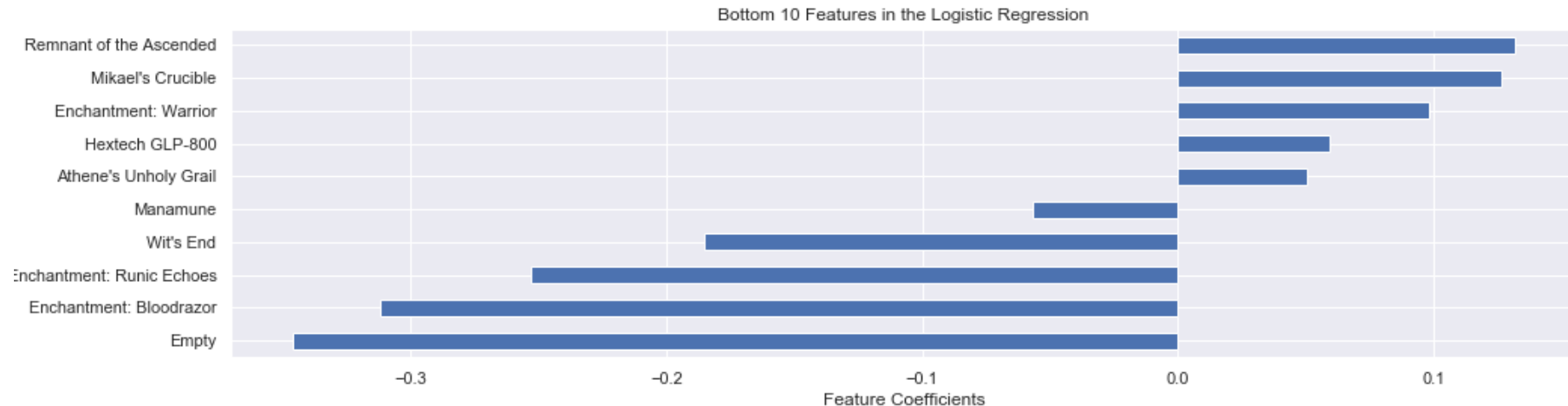
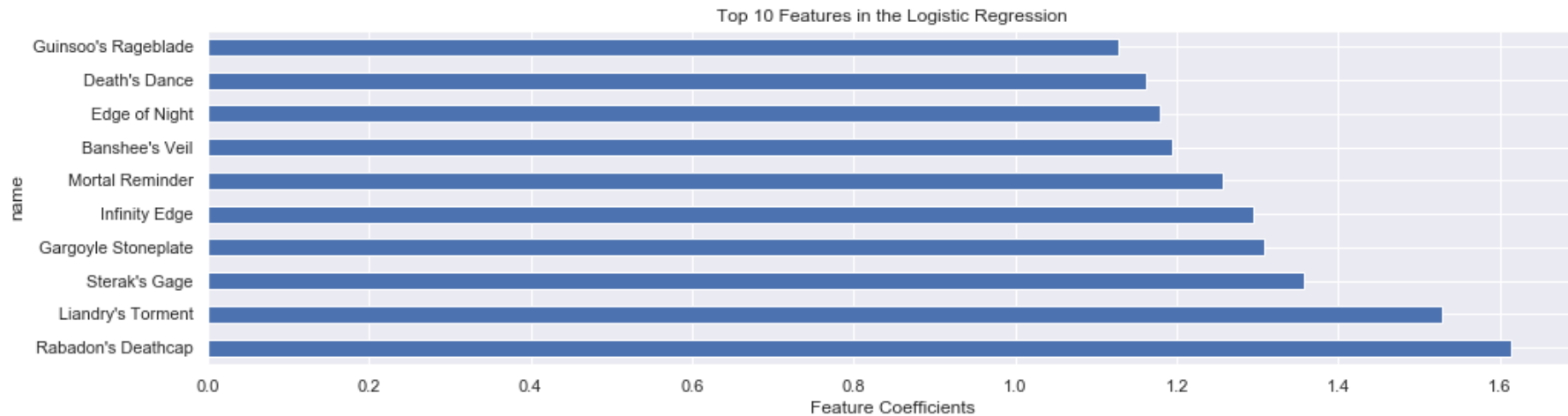
# Machine Learning: Logistic Regression

- Feature Coefficients
  - Higher numbers == Higher chance of winning
  - Lower numbers == Lower chance of winning
- A few coefficients below 0...
- Only a few very high
- Potential Anomalies!



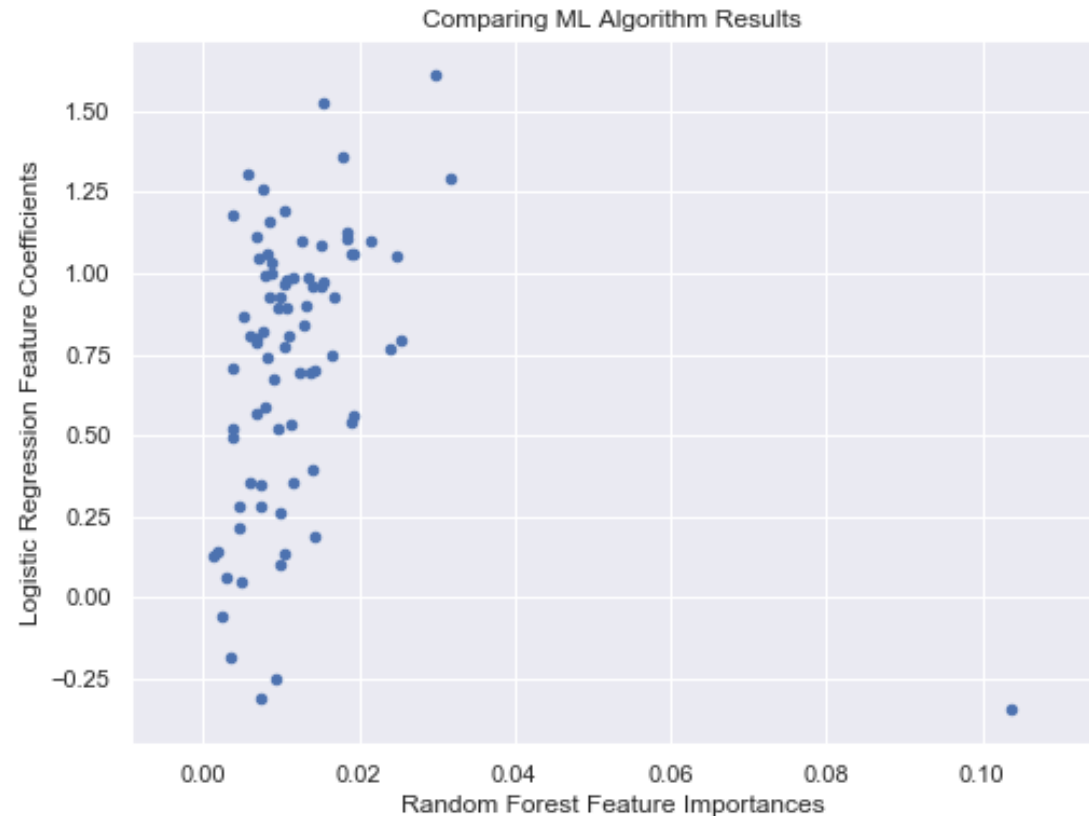
# Machine Learning: Logistic Regression

- Which items have good coefficients? Bad?



# Machine Learning: Comparing Models

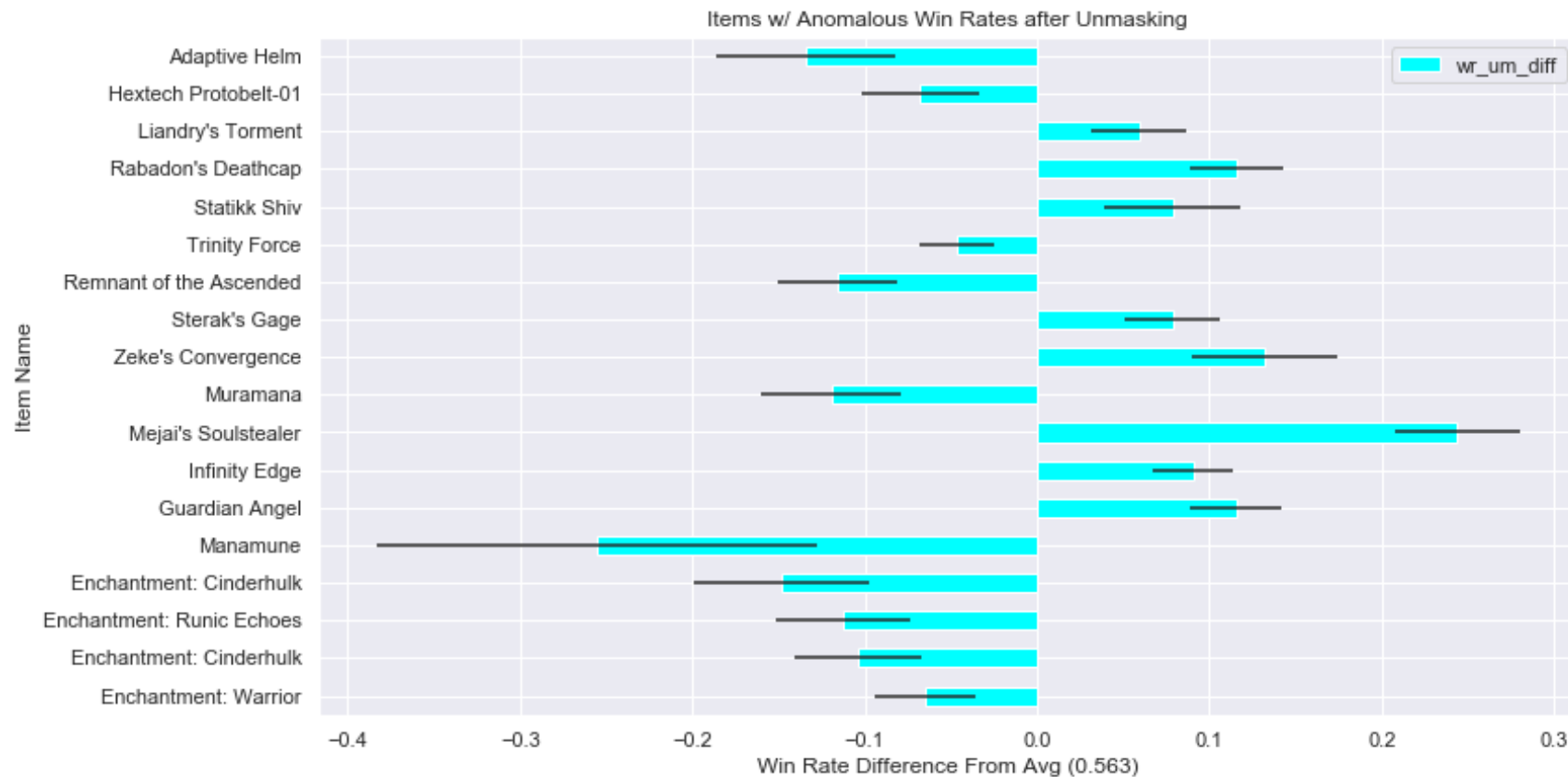
- Do the models agree on what items are important?
- Almost completely!
- Only exception is the Empty item slot



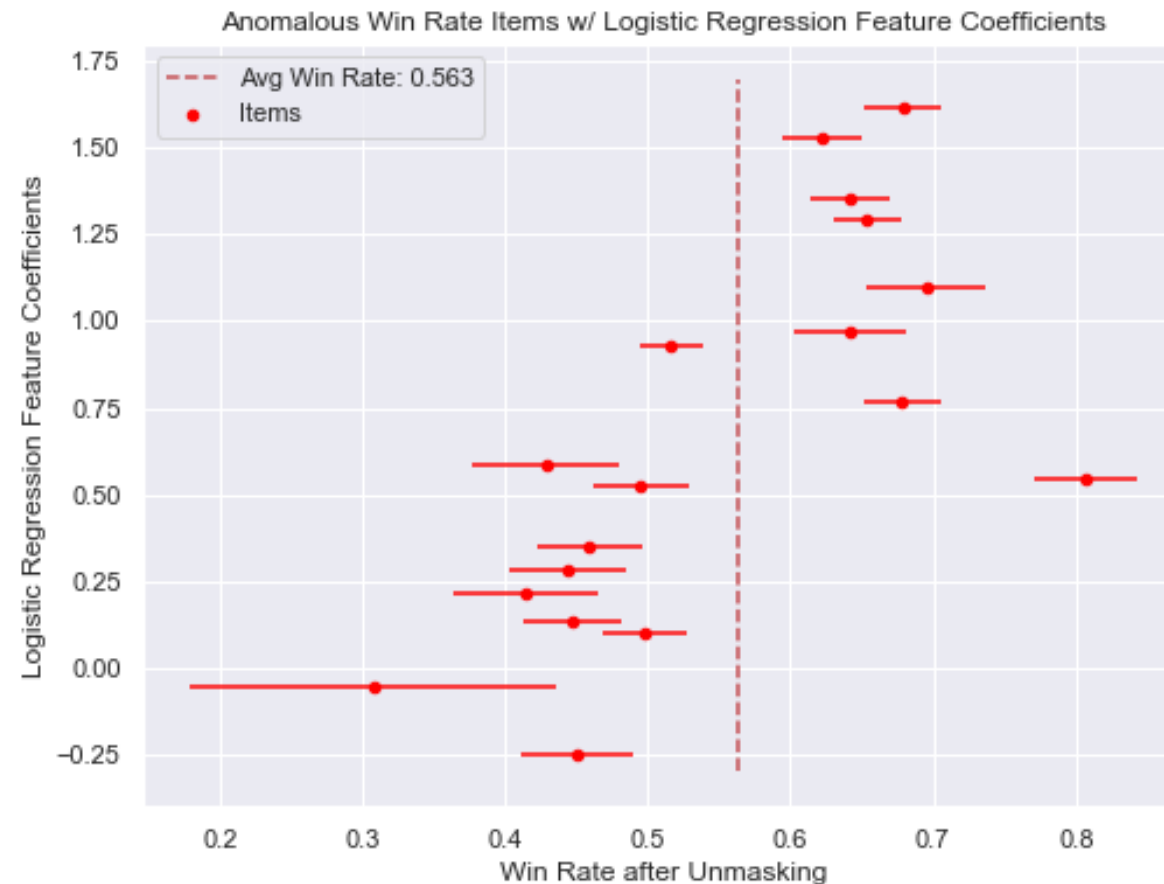


# Machine Learning: Compare to my Statistics

- Recall this plot



# Machine Learning: Compare to my Statistics



# Machine Learning: Compare to my Statistics



# Machine Learning: Bias Problems?

- The 0 column
  - Good for predicting a game's outcome
  - Not technically an item
- The Logistic Regression intercept
  - With an intercept, feeding the model all 0s doesn't return 50/50
  - It should



VS

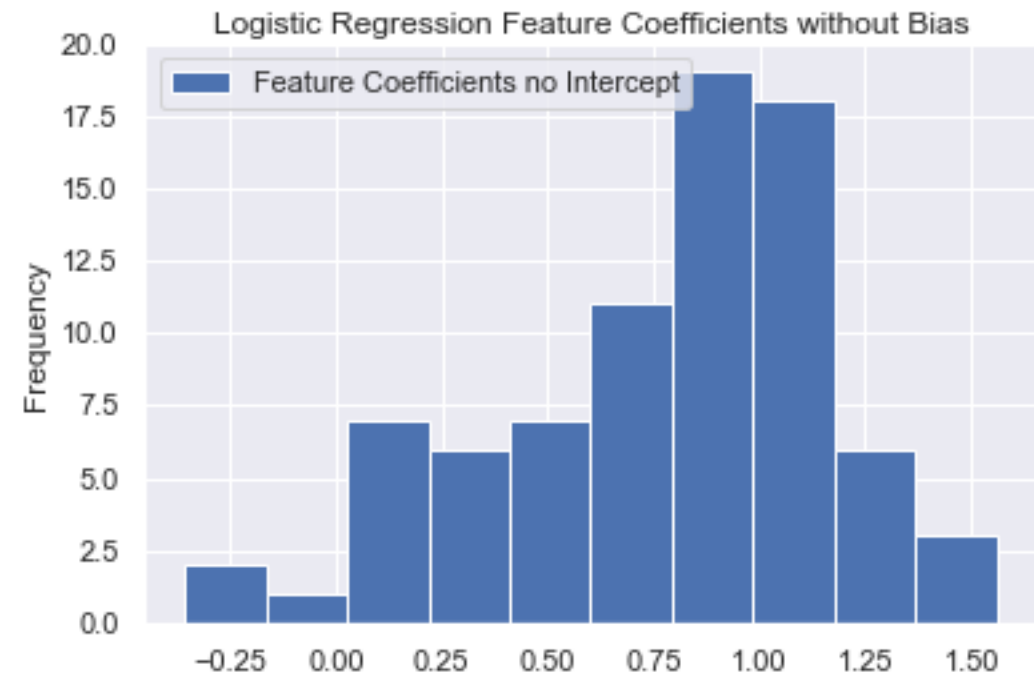
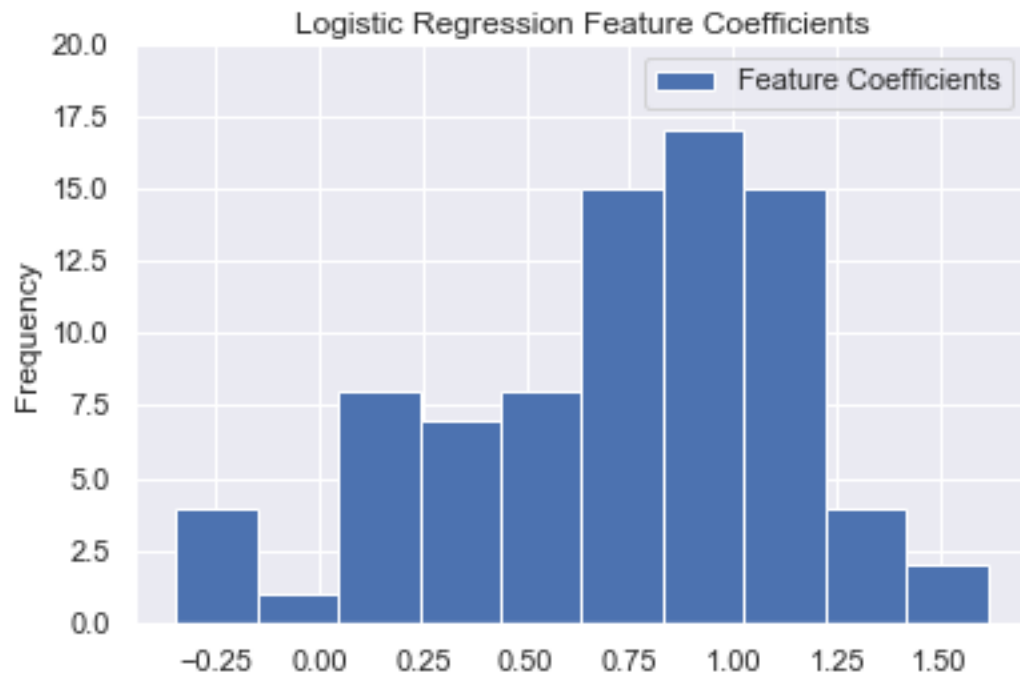


# Machine Learning: Logistic Regression 2.0

- Remove the 0 column from the data
- Don't fit an intercept
- Training accuracy: 0.898
- Test accuracy: 0.881
- Only slightly worse at predicting match results
- Feature coefficients should be slightly more useful now
- Grid search yielded the default hyperparameters

# Machine Learning: Logistic Regression 2.0

- How do the feature coefficients compare to the first LR?
- Very similar



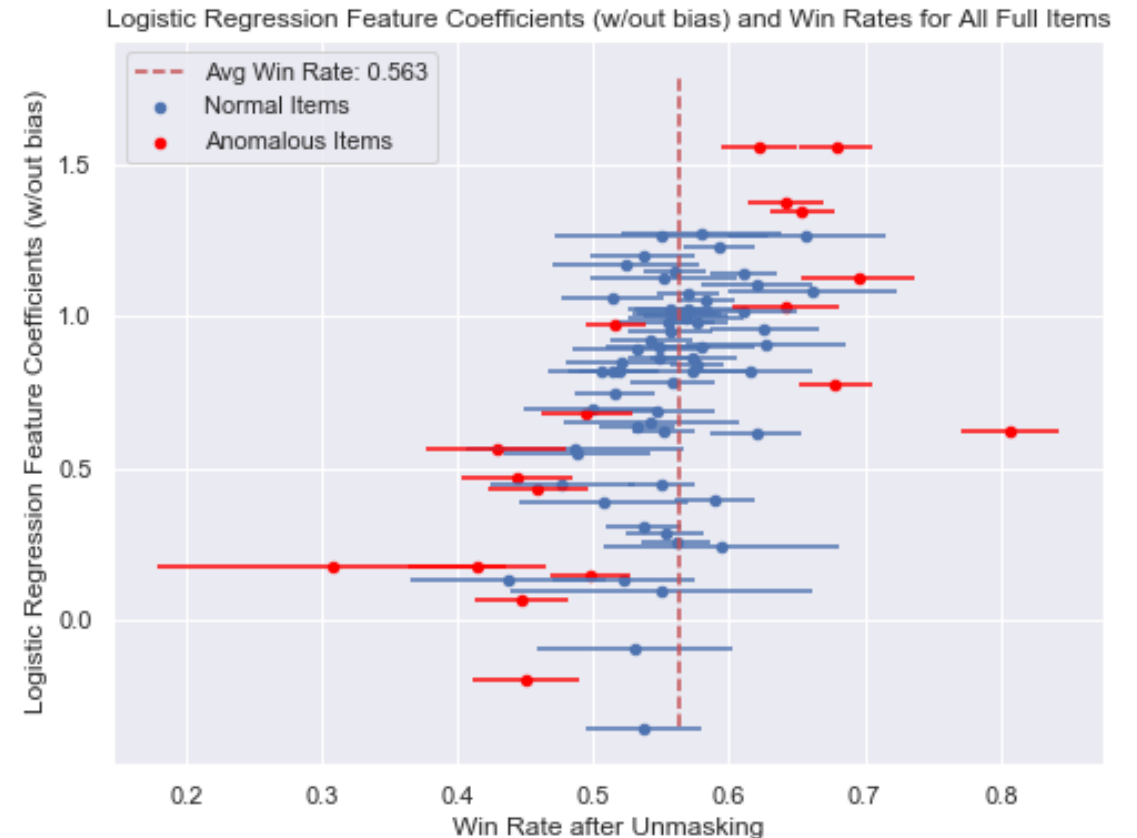


# Machine Learning: Compare to my Statistics 2.0

- With Intercept and 0 column



- Without Intercept and 0 column



# Outline

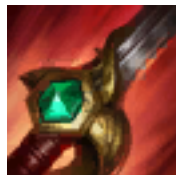
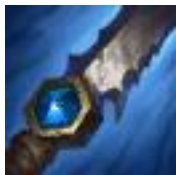
- Crash Course: League of Legends
- Problem Statement
- Data
- Statistics
- Machine Learning
- **Conclusions**

# What should I recommend?

- Statistical analyses
  - Avg win rate was  $> 1.96$  std dev's away from the item's win rate
- Logistic Regression
  - Top 10 / Bottom 10 Feature Coefficients
  - Unusually high / low impact on the game's result
- That intersection – strong rebalance recommendation
- Their union minus the intersection – moderate rebalance recommendation

# Recommendations: Items that are Too Weak

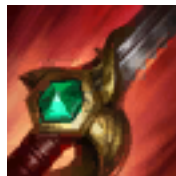
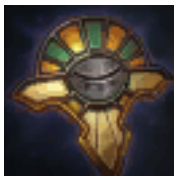
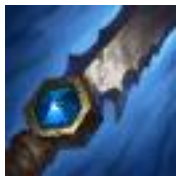
- Flagged by both analyses
  - Runic Echoes (blue)
  - Remnant of the Ascended
  - Warrior (blue)
  - Manamune
  - Cinderhulk (red)



- Flagged by one of the analyses
- Logistic Regression
  - Bloodrazor (red)
  - Wit's End
  - Mikael's Crucible
  - Hextech GLP-800
  - Athene's Unholy Grail
- Win Rate Statistics
  - Adaptive Helm
  - Hextech Protobelt-01
  - Trinity Force
  - Muramana
  - Cinderhulk (blue)

# Recommendations: Items that are Too Weak 2.0

- Flagged by both analyses
  - Runic Echoes (blue)
  - Remnant of the Ascended
  - Warrior (blue)
  - Manamune
  - Cinderhulk (red)



- Flagged by one of the analyses
- Logistic Regression
  - Bloodrazor (red)
  - Wit's End
  - Mikael's Crucible
  - Hextech GLP-800
  - Athene's Unholy Grail
- Win Rate Statistics
  - Adaptive Helm
  - Hextech Protobelt-01
  - Trinity Force
  - Muramana
  - Cinderhulk (blue)

# Recommendations: Items that are Too Strong

- Flagged by both analyses

- Rabadon's Deathcap
- Liandry's Torment
- Sterak's Gage
- Infinity Edge



- Flagged by one of the analyses

- Logistic Regression

- Mortal Reminder
- Edge of Night
- Gargoyle Stoneplate
- Guinsoo's Rageblade
- Thornmail
- Frozen Mallet

- Win Rate Statistics

- Statikk Shiv
- Zeke's Convergence
- Mejai's Soulstealer
- Guardian Angel



# Recommendations: Items that are Too Strong 2.0

- Flagged by both analyses

- Rabadon's Deathcap
- Liandry's Torment
- Sterak's Gage
- Infinity Edge



- Flagged by one of the analyses

- Logistic Regression

- Mortal Reminder
- Edge of Night
- Gargoyle Stoneplate
- Guinsoo's Rageblade
- Thornmail
- Frozen Mallet

- Win Rate Statistics

- Statikk Shiv
- Zeke's Convergence
- Mejai's Soulstealer
- Guardian Angel

# Recommendations: Final

- Make stronger

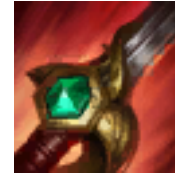
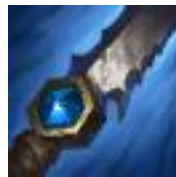
- Runic Echoes (blue)
- Remnant of the Ascended
- Manamune
- Cinderhulk (red)

- Make weaker

- Rabadon's Deathcap
- Liandry's Torment
- Sterak's Gage
- Infinity Edge

- Consider making stronger

- Wit's End
- Mikael's Crucible
- Hextech GLP-800
- Athene's Unholy Grail
- Adaptive Helm
- Hextech Protobelt-01
- Trinity Force
- Muramana



- Consider making weaker

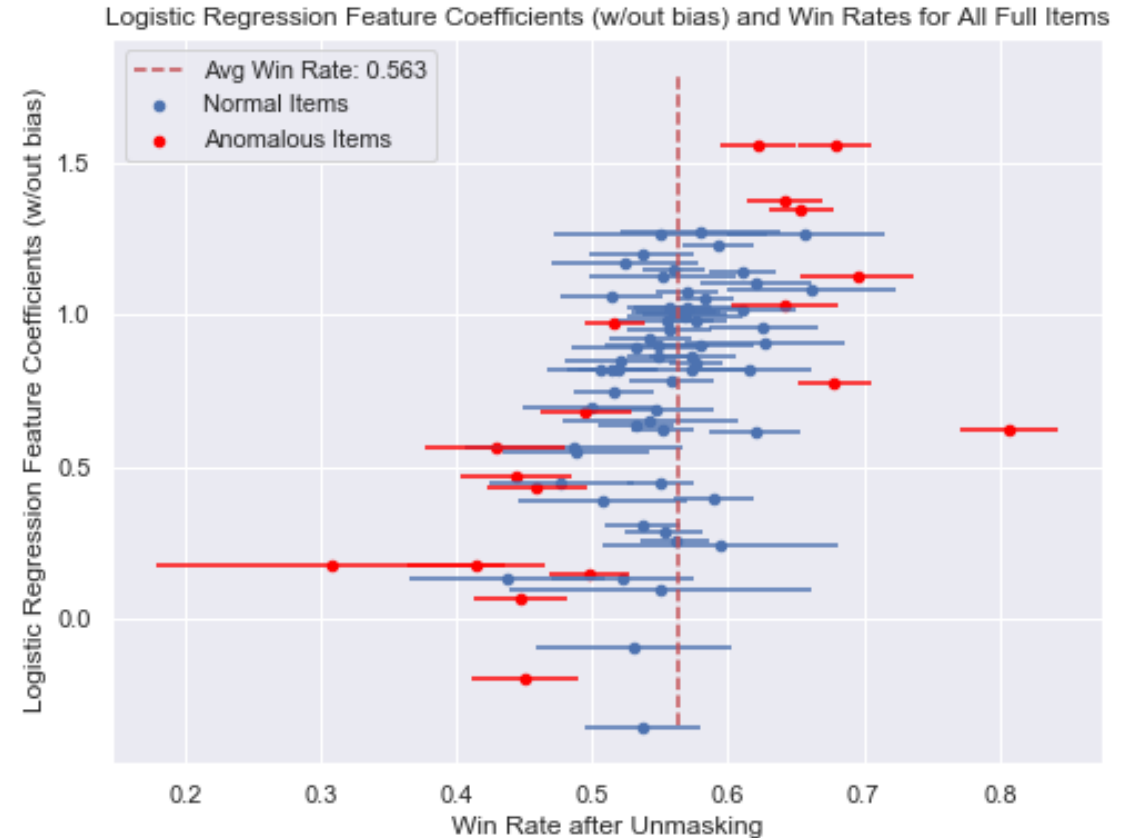
- Mortal Reminder
- Edge of Night
- Gargoyle Stoneplate
- Guinsoo's Rageblade
- Statikk Shiv
- Zeke's Convergence
- Mejai's Soulstealer
- Guardian Angel

# Room for Improvement

- Try to change items based on popularity (pick rate)
- Timing data
- Consider the meta
- Model as a whole should consider champions as well
  - This adds enormously to the work I would need to do
- Different ML models?

# Conclusions

- Wanted to find a way to recommend item changes
- Statistical analysis of win rates
- Machine learning models to find high- and low- impact items
- Found 4 items to buff, 4 items to nerf



# Backup Slides

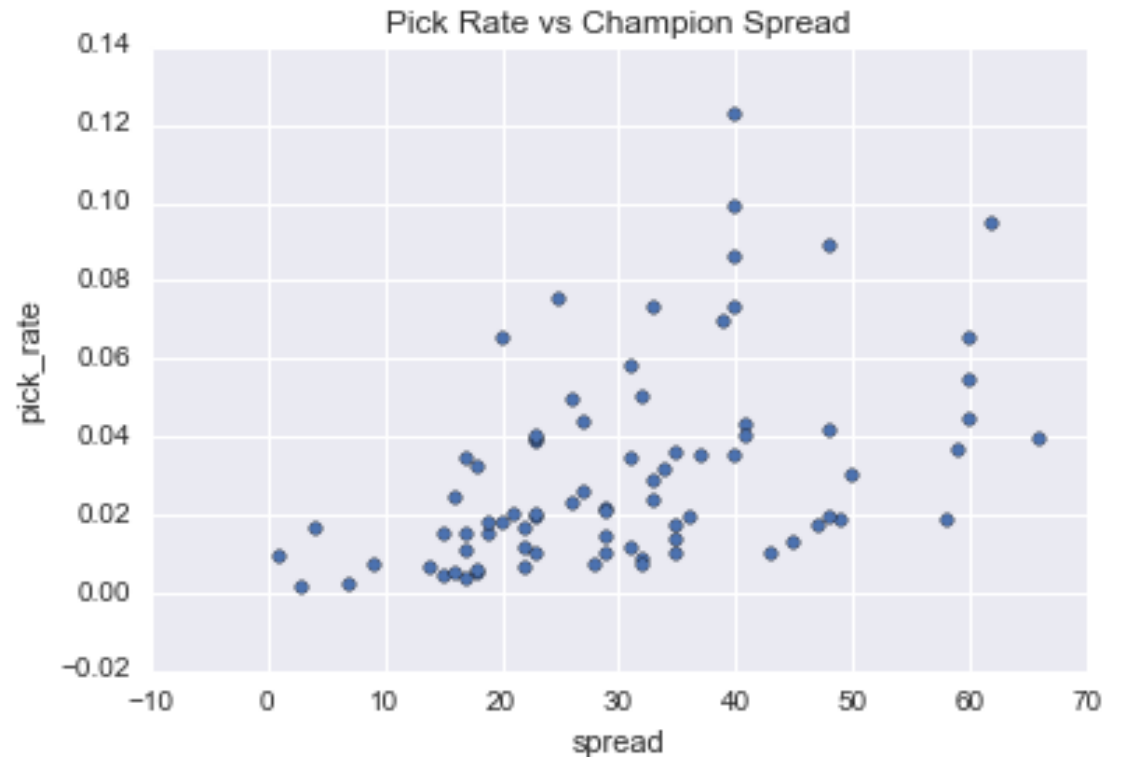
# Wrinkle About Tags

- Original Tag List
  - `u'[Fighter, Assassin]'`
- Regex
  - `df['tags'] = [re.findall('[A-Za-z]\w*', df.loc[champion, 'tags'])  
for champion in df.index]`
- Final Tag List
  - `[u'Fighter', u'Assassin']`



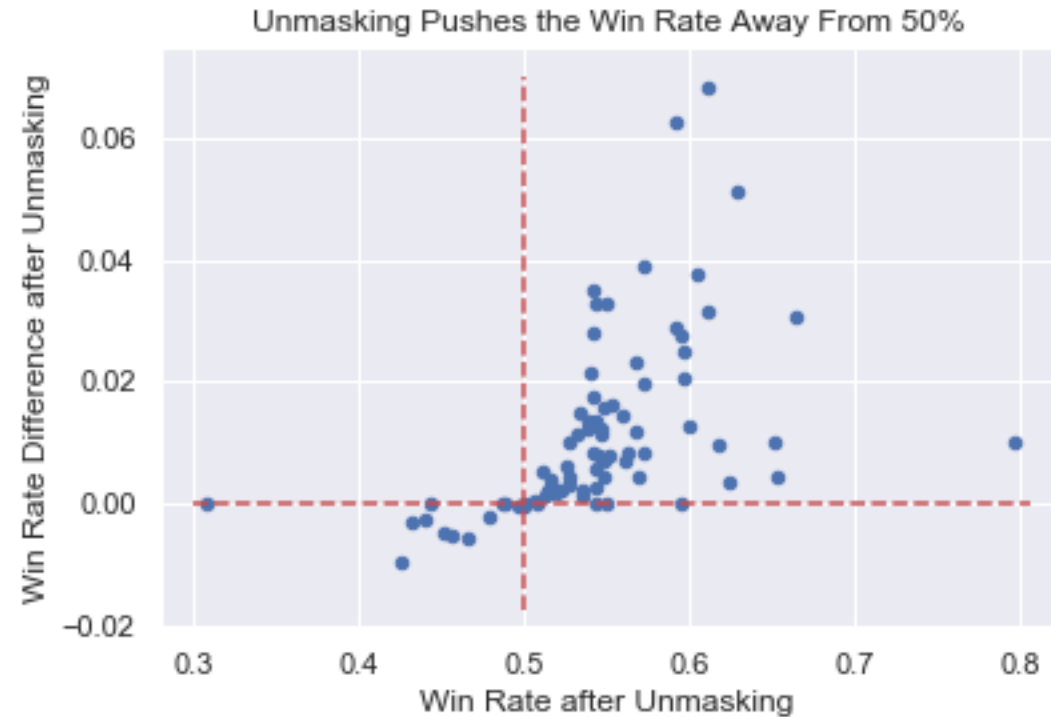
# Item Stats: Champion Spread

- How many unique champions buy an item
  - 141 champions total
  - Low spread = low flexibility
  - High spread = excessive flexibility



# Statistics: Unmasked Win Rates

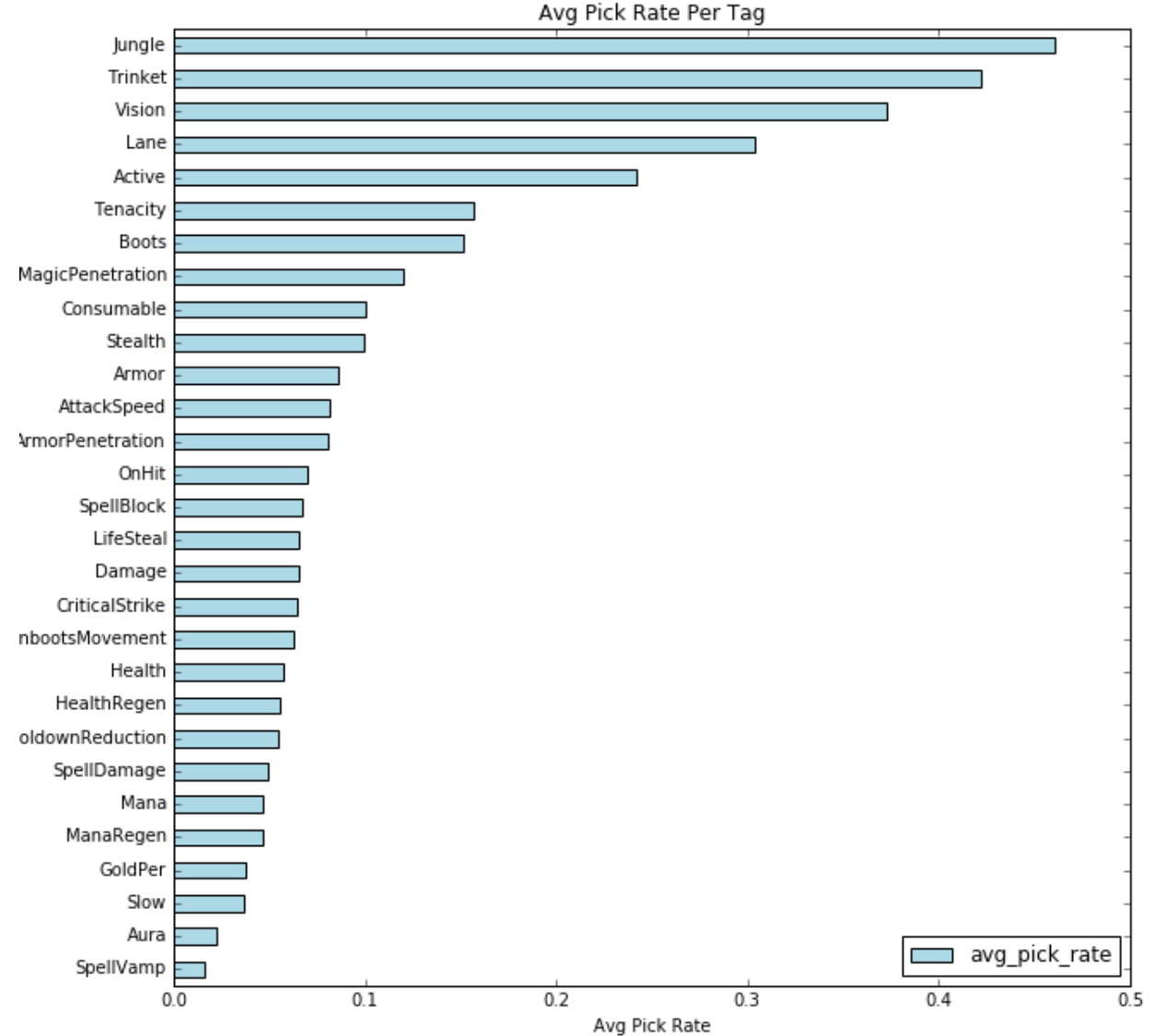
- I figured that this masking pushes win rates towards 0.50
- Unmasking pushes win rates away from 0.50



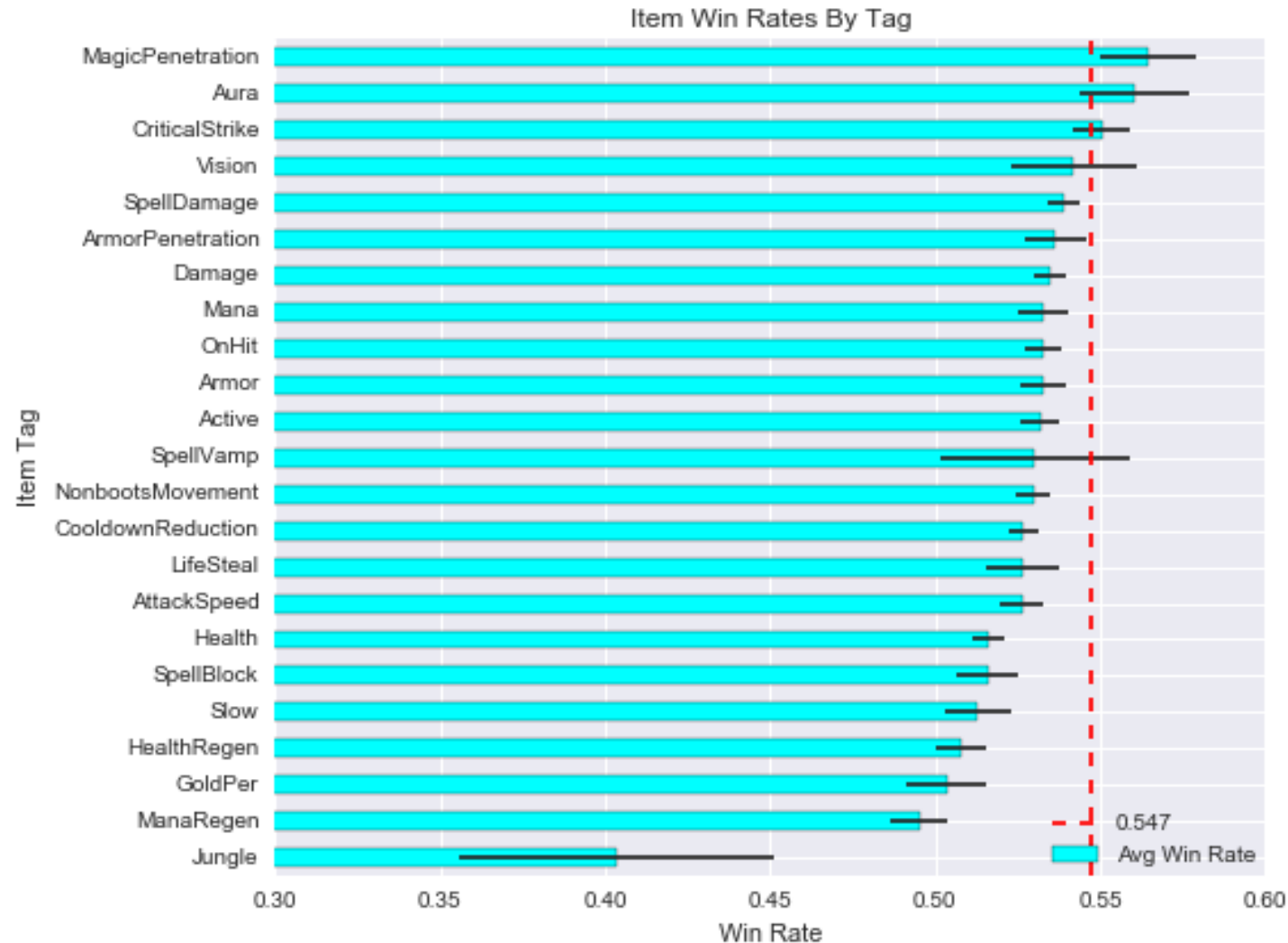
# Aggregate by Item Tags

- Items can be grouped into sets based on the bonuses they give
- Example tags:
  - Damage
  - SpellDamage
  - Health
  - Armor
  - SpellBlock
  - AttackSpeed

# Pick Rate vs Tag



# Win Rate Per Tag



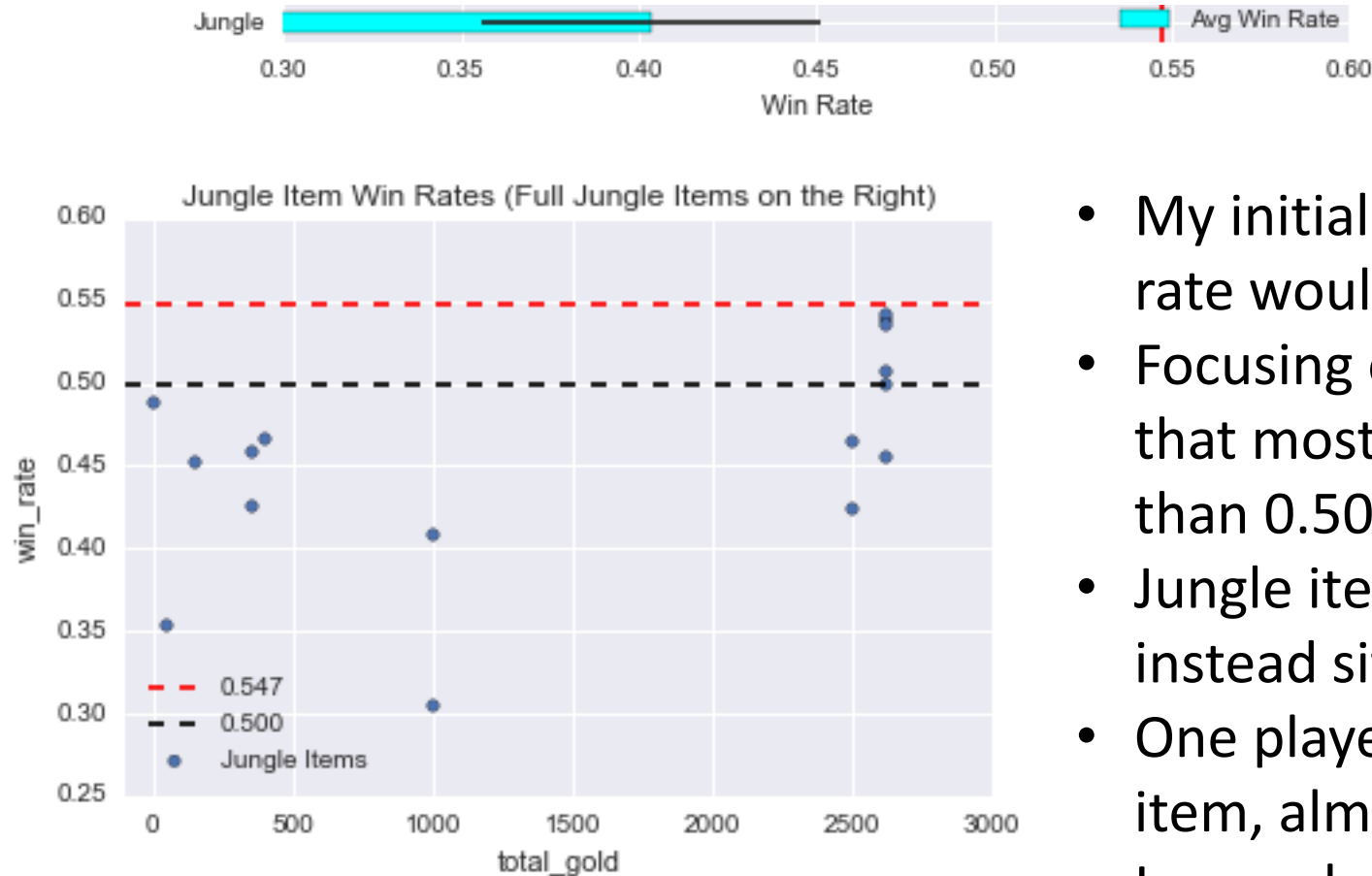
- Average win rate for all items of the tag
- Std deviation of the win rate for the tags' items

# Summoner Spells (Smite Specifically)

- Flash
  - Ghost
  - Heal
  - Ignite
  - Teleport
  - Exhaust
  - Smite
  - Cleanse
  - Barrier
- Smite is required for Junglers
  - Jungle items cannot be purchased if the summoner does not have smite
  - Jungle items grant bonus gold and experience when jungle monsters are slain
    - Jungling is not viable without a jungle item



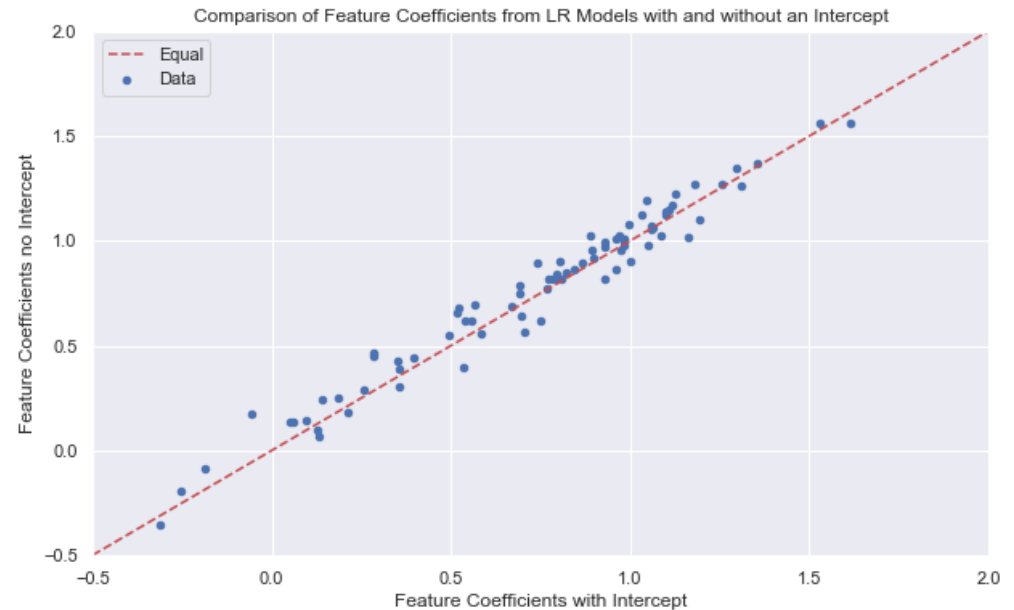
# Statistics: Jungle Items



- My initial thought was that the avg win rate would be 0.50
- Focusing on full items, it made sense that most win rates would be higher than 0.50
- Jungle items do not follow this trend, but instead sit around 0.50
- One player per team builds a jungle item, almost always first
- Incomplete jungle items are uncommon.

# Machine Learning: Logistic Regression 2.0

- How do the feature coefficients compare to the first LR?
- Very similar



# Machine Learning: Item Costs

- Stronger / more expensive items should have higher impact
- Normalize feature coefficients by item cost

