Contents

Aim

This project aims to investigate a set of League of Legends data covering the statistics of Challenger ranked players over January 2022, evaluating the relationships between various configurations of summoner spells and other key gameplay elements such as KDA and vision score. Given said gameplay statistics, can we accurately predict a Challenger ranked player's selected summoner spell configuration using a supervised machine learning model?

The results of this project would be of particular interest to game hobbyists/ casual players of League of Legends, as well as e-sports enthusiasts and professional players involved in the game.

Comparing summoner spell configurations with a wide range of in-game statistics yields a broader range of beneficial information for the audience of this project, providing further insight into summoner spell selection. Particular roles in League of Legends are primarily focused on achieving different objectives in game, objectives which may be influenced by a player's summoner spells. Thus, highlighting which summoner spell configurations correlate with specific in-game statistics can provide a great deal of insight for professional players, or even dedicated casual players, who are trying to fine tune their performance.

Datasets

An edited version of the following three datasets were used in our investigation; “LoL Challenger Soloq Data (Jan, Kr-Na-Euw).” [1. For reference]. The three datasets were sourced from different regions (Korea, North America and Western Europe) and consisted of player data from matches played in January 2022 at the Challenger rank of League of Legends. For each of the matches played in a single randomly chosen player within that match, along with their relevant individual match statistics, were inputted into their respective regional datasets. Furthermore, the game ID of each match was used to ensure that no two players were associated with the same match, minimising the chances of any duplicate entries appearing. Between the three datasets there were 17228 recorded player end-of-game statistics.The compiled data set was comprised of 20 initial columns, all of which relate to individual statistics of that player within the match.

Description of features in the data set:

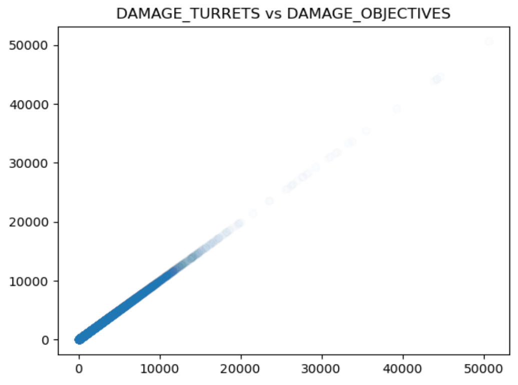
Feature	Description
d_spell	summoner spell on d key
f_spell	summoner spell on f key
champion	champion being played
side	side of map player is on red/blue
role	role being played out of the 5
assists	number of assists in match
damage_objectives	damage to objectives
damage_building	damage to buildings
damage_turrets	damage to turrets
deaths	deaths in game
gold_earned	gold earned in game
kda	k/d/a ratio in game
kills	kills in game
level	level in game
time_cc	time crowd controlling others
damage_total	total damage in game
damage_taken	total damage taken in game
minions_killed	total minions killed in game
turret_kills	turret kills in game
vision_score	vision score in game

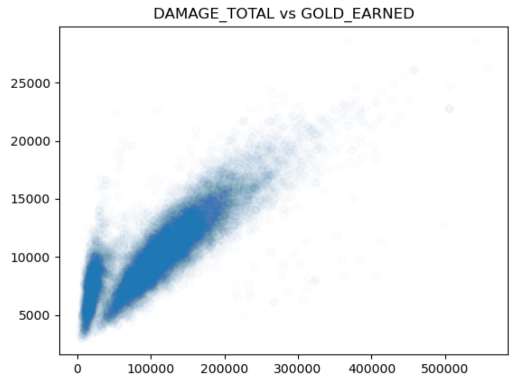
Pre-processing and Data Wrangling

Prior to undergoing a thorough analysis of the dataset we confirmed that the data set contained the features described by the author, and that all entries were appropriately formatted and readable. Overall the data quality was of a high standard, other than numerous instances of missing values. We also observed that role seemed to be compressed from the usual five standard roles into ‘Other’ and ‘TopLane_Jungle’. Additionally, minions_killed was categorised into ‘Many’ and ‘Few’. Note, all three datasets here were stored as separate csv files, and were read into pandas dataframes for the following pre-processing and data wrangling steps.

We looked for any significant variances in the distributions of column values between the datasets and concluded that the differences were not significant enough to have any impact on our supervised learning model. This allowed us to combine the datasets, adding an additional column to denote the region so that regional differences were preserved.

As a preliminary analysis, we generated scatter plots and calculated Pearson correlation values for all columns in the dataset. This allowed us to determine how to approach the dataset for our supervised learning task. Of particular note, we observed a one-to-one relationship between damage_objectives and damage_turret and concluded that data was missing at random.

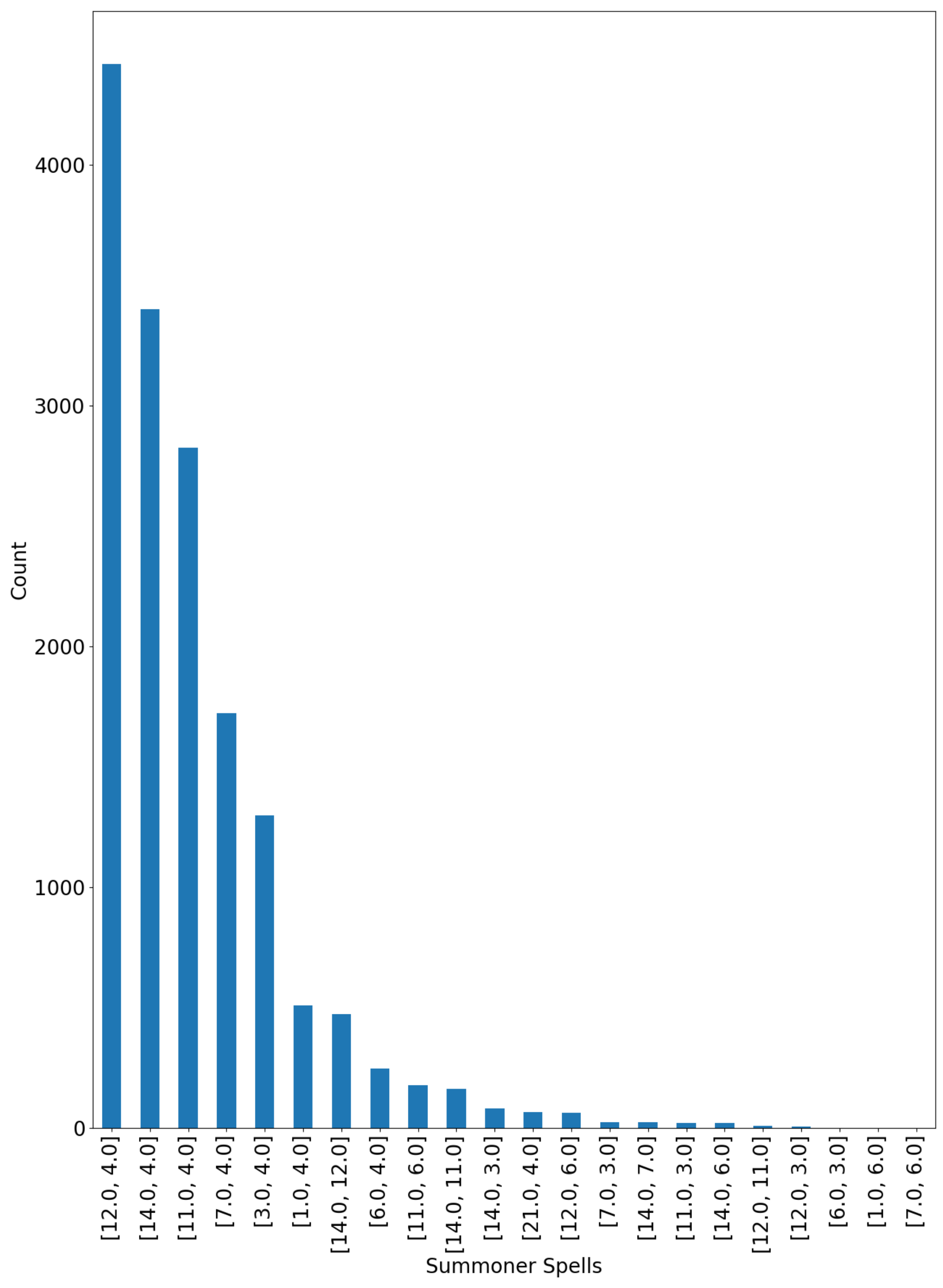


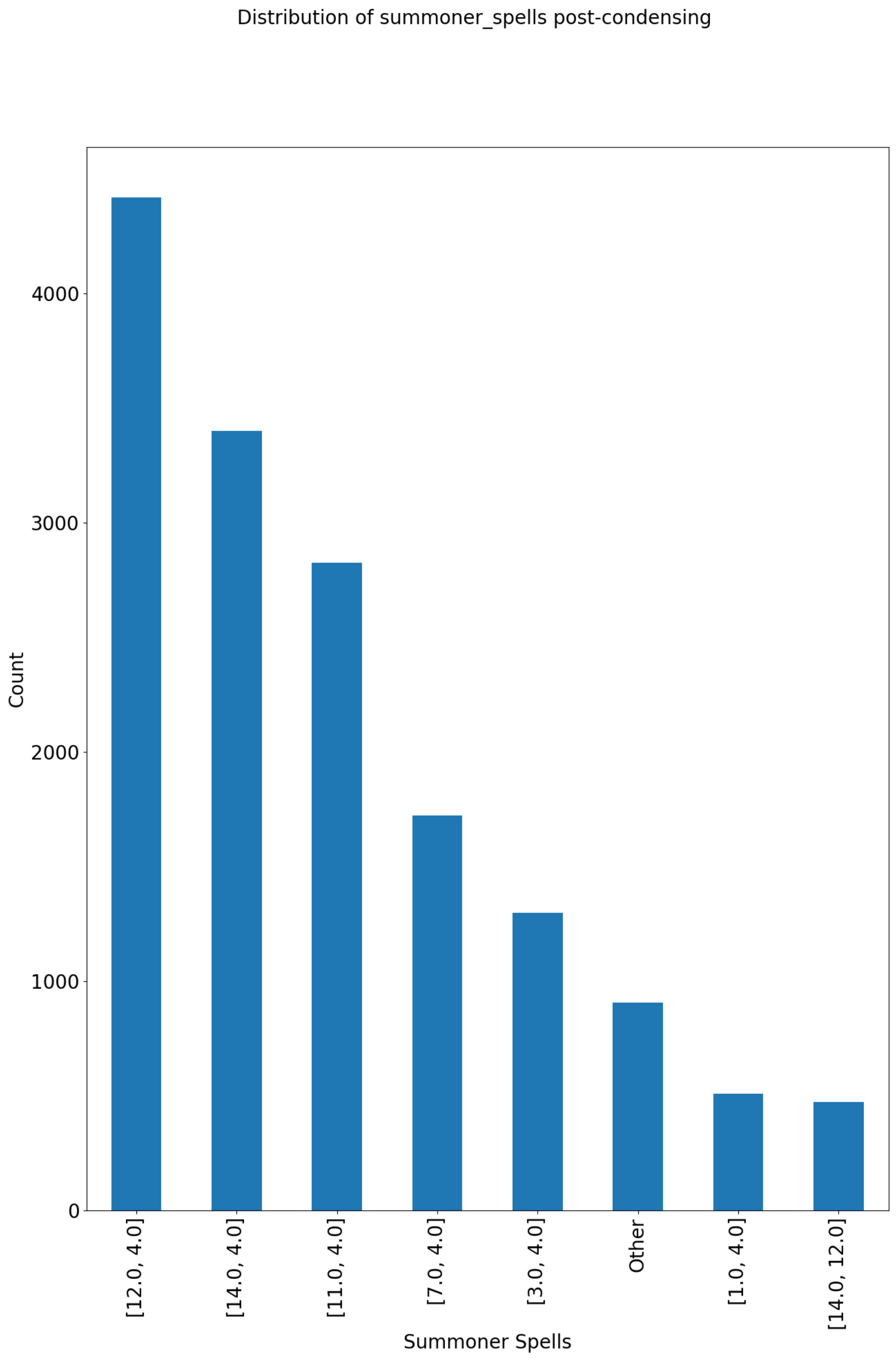


The next step was to impute on the columns that we knew could be determined with certainty using other features: KDA, kills, deaths, assists, damage_objectives and damage_turrets. This gained us 4508 additional values to work with. The calculations of kills, deaths and assists caused precision issues so we rounded them to rectify this. For example, there were 125 unique values in ‘kills’ pre-rounding, compared to the 36 unique values post-rounding.

Further, the d_spell and f_spell columns were merged into a new ‘summoner_spells’ column and formatted into a standard orientation to account for duplicate configurations, seeing as the exact orientation of a specific combination of summoner spells is entirely dependent on arbitrary player preferences. With summoner_spell being the main comparative factor of our project, we decided that some spell configurations were so uncommon that they wouldn’t be useful to have as separate entries, so all spell configurations that constituted less than 2.5% of the total of data entries were grouped together into an ‘Other’ section, as seen in the figures below.

Distribution of summoner_spells pre-condensing




Distribution of summoner_spells post-condensing
Count
4000
3000
2000
1000
0
[12.0, 4.0]
[14.0, 4.0]
[11.0, 4.0]
[7.0, 4.0]
[3.0, 4.0]
Other
[1.0, 4.0]
[14.0, 12.0]
Summoner Spells

We dropped columns d_spell, f_spell and damage_turrets from the dataset as they were now obsolete given we had now stored their data elsewhere. Additionally, we dropped rows with missing values in champion since we thought it would be unrealistic to impute on. In the same vein, we dropped rows with missing values in summoner_spells as it was our target variable. Using domain knowledge we then decided to remove the side column as the ‘side’ of the player was entirely arbitrary and had no implications on summoner spell selection. Lastly, we dropped duplicates as it was unlikely that they occurred naturally within the dataset.

We wanted to directly control how the data was encoded, rather than relying on imported libraries, so we wrote our own encoding functions. ‘Summoner_spells’ was encoded based on the frequency of labels (highest to lowest), ‘champion’ was encoded alphabetically, and ‘regions’, ‘minions_killed’ and ‘role’ were encoded arbitrarily.

All numerical data (both discrete and continuous) were binned using three uniform width bins. We originally excluded outliers from the calculations of bin widths, however this skewed many data points towards the lower bins leaving little room for discrimination between values. Quantile bins were also considered, however due to the dataset being heavily imbalanced this was not feasible. The use of three bins, as opposed to other options, was decided upon across all columns as we felt it was separate enough to distinguish between low, medium and high values without it being vulnerable to data noise. Additionally it maintained consistency between columns and would lead to more accurate imputations later on. Binning also increased the time efficiency of fitting our supervised learning models, more specifically K-Fold cross validation run-time.

During the model evaluation stage we decided to use a simple imputation strategy by imputing on nan values by mode. This allowed us to retain more records for model training at the expense of adding bias to statistical measurements such as standard deviation. Alternatively, we could’ve used multivariate strategies, such as developing models to predict the missing values, but we concluded that it wasn’t necessary.

Analysis Methods

From the beginning we decided to use a training-test split. A training-test split allows you to separate a dataset so that you can both build the model with one set, and evaluate its usefulness and performance with the other. We split the 14410 records on a 80:20 split, leaving 11528 records for training and 2882 for testing. We thought this provided a sufficient amount of data for both operations.

In our preliminary analysis of the dataset it was abundantly clear that there were strong non-linear relationships between some of the variables. As our target variable was also categorical, we could not use analytical tools such as regressions or Pearson correlation for feature selection. Instead, we opted to rank the features via normalised mutual information. Normalised mutual information allows us to both detect non-linear relationships and measure correlation between discrete variables. An issue with this however was that it was sensitive to discrete numerical values, so binning was necessary. On top of this, the results varied greatly depending on the type of binning. Lastly, as a feature selection method, mutual information is not good at accounting for inter dependencies between variables. An alternative method for feature selection is Chi-square, however we found NMI easier to implement.

Feature	NMI with summoner_spell
champion	0.64788
role	0.54813
minions_killed	0.50805
binned_vision_score	0.26387
binned_damage_building	0.25089
binned_damage_taken	0.12033
binned_time_cc	0.11506
binned_damage_objectives	0.09512
binned_assists	0.05464
binned_kills	0.05121
binned_damage_total	0.04912
binned_gold_earned	0.04273
binned_level	0.03954

binned_kda	0.02547
region	0.00882
binned_deaths	0.00425

Since our target variable was categorical, we needed to use a classification supervised ML model. To evaluate the most suitable classification model, we used the training set to perform a K-Fold cross validation (5-Fold) on Decision Tree and Knn whilst tuning selected hyperparameters iteratively. Both models were evaluated using one to five selected features (based on highest NMI with target variable). Additionally, for the Decision Tree model we adjusted the criterion for measuring the quality of the split between 'entropy' or 'gini'. As for the Knn model, we iterated through three to seven closest neighbours. We allowed for imputation by the training set mode on missing values in the training set. The training set mode was used to avoid data leakage. We selected the model based on accuracy for simplicity; other possible metrics for evaluating include recall, precision and F1 scores.

Decision Tree Model Accuracy	**One Feature**	**Two Features**	**Three Feature**	**Four Features**	**Five Features**
Entropy Based	0.72719	0.76336	0.78877	0.78851	0.78713
Gini Based	0.72719	0.76336	0.78877	0.78851	0.78721

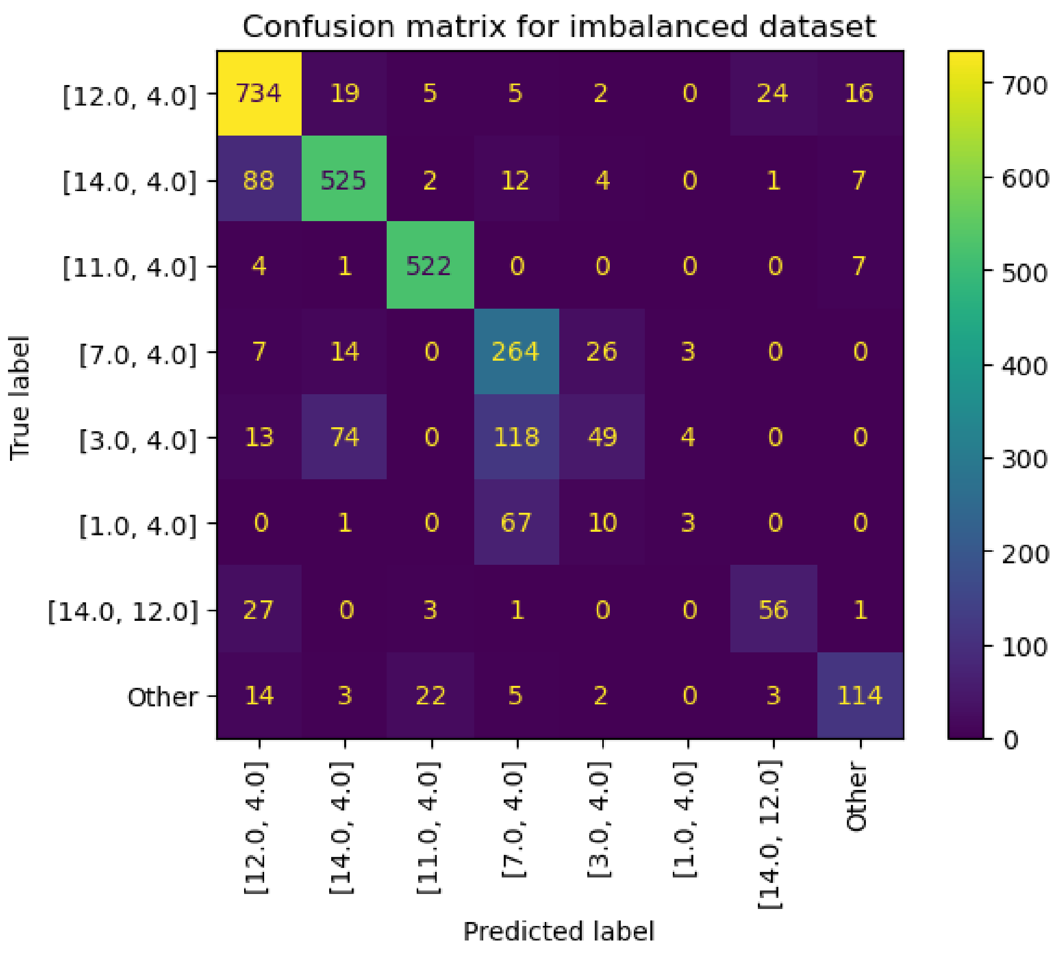
KNN Accuracy	**One Feature**	**Two Features**	**Three Feature**	**Four Features**	**Five Features**
3NN	0.68121	0.71981	0.74895	0.75190	0.74496
4NN	0.69387	0.72666	0.76795	0.76795	0.76231
5NN	0.70376	0.73716	0.77255	0.77211	0.76778
6NN	0.70437	0.73855	0.77099	0.77324	0.76804
7NN	0.70134	0.74072	0.77177	0.76960	0.76570

Depending on random sample variation, the best model according to K-Fold CV was a Decision Tree with three or four features with an accuracy of ~0.79. Note that the difference between entropy and gini based was virtually indistinguishable. We allowed for multiway splits and unlimited depth, however the algorithm decided against multiway splits in favour of binary splits. Forcing multiway splits and/ or limiting depth yielded worse results.

Finally, we fitted the decided best model on the whole training set and evaluated the results on the test set, producing a decision tree and confusion matrix as well as a range of evaluative statistics.

Accuracy (Weighted)	0.78661
Precision (Weighted)	0.76982
F1 (Weighted)	0.76437

	[12, 4]	**[14, 4]**	**[11, 4]**	**[7, 4]**	**[3, 4]**	**[1, 4]**	**[14, 12]**	**Other**
Subsect Accuracy	0.912	0.822	0.978	0.841	0.190	0.037	0.636	0.699
Subset Precision	0.828	0.824	0.942	0.559	0.527	0.3	0.667	0.786
Subset F1	0.868	0.823	0.960	0.672	0.279	0.066	0.651	0.740



Glossary of spells:

1 is cleanse

3 is exhaust

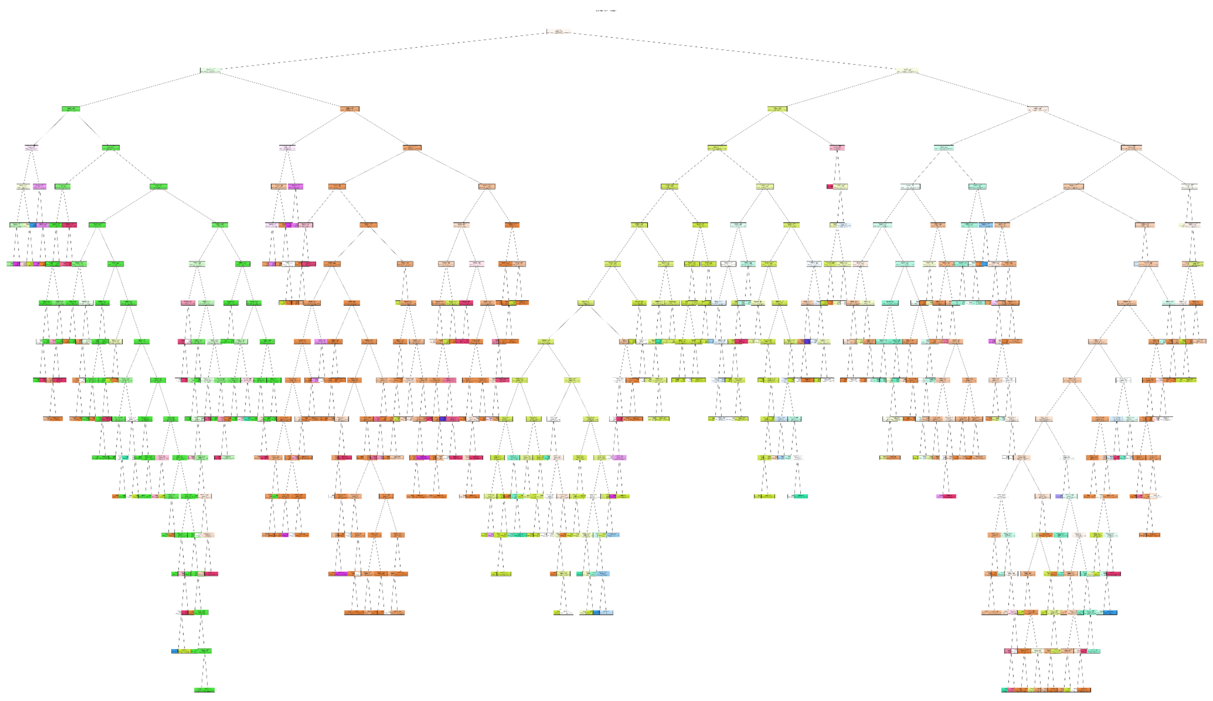
4 is flash

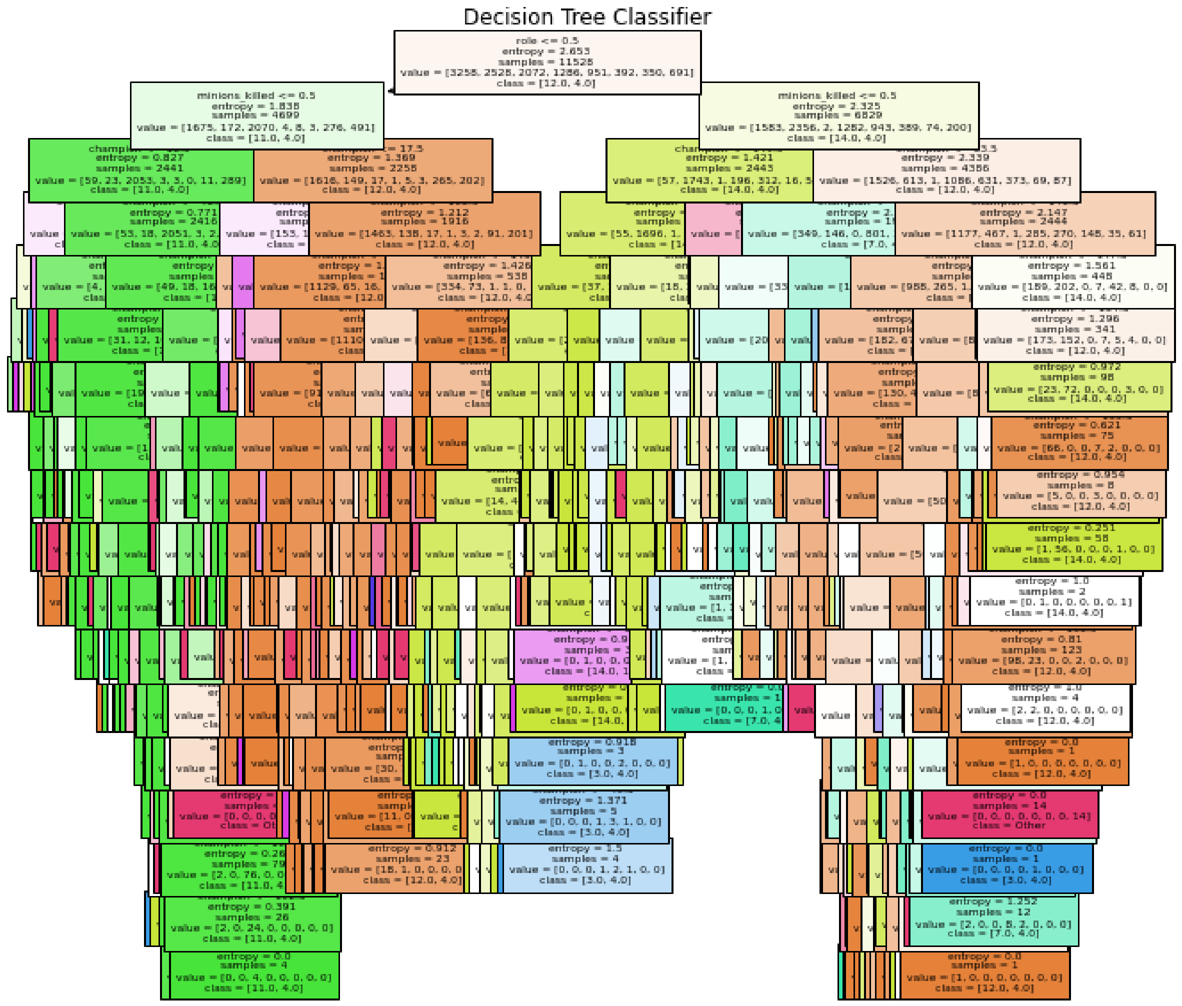
7 is heal

11 is smite

12 is teleport

14 is ignite




Decision Tree Classifier
role <= 0.5
entropy = 2.653
samples = 11528
value = [3258, 2528, 2072, 1286, 951, 392, 350, 691]
class = [12.0, 4.0]
minions_killed <= 0.5
entropy = 1.838
samples = 4699
value = [1675, 172, 2070, 4, 8, 3, 276, 491]
class = [11.0, 4.0]
minions_killed <= 0.5
entropy = 2.325
samples = 6829
value = [1583, 2356, 2, 1282, 943, 389, 74, 200]
class = [14.0, 4.0]

Discussion

Before addressing the results, it is important to acknowledge the over-representation of flash (4) in the dataset. Flash has always been popular due to the various movement benefits it provides so these results are not unexpected.

Our feature selection methods led us to using champion, role and minions_killed for our final model. Early on in our investigation we hypothesised that champion and role selection would be the most important features for our supervised learning task, so we were not surprised by this. Even though role and minions_killed had both been simplified, potentially limiting their utility, they were still sufficiently indicative of summoner spell selection.

Overall the model performed relatively well with a weighted accuracy, precision and F1 of 79%, 77% and 76% respectively. For the smite (11) and flash combination specifically, it achieved an accuracy, presion and F1 of 98%, 94% and 96% respectively. This is quite impressive given that the model only used three features from the dataset. The Jungle role, and thus champions associated with jungle, are known to predominantly use smite and flash. The model was able to predict this association at a stronger level than we expected.

Despite this, there are obvious areas where the model underperforms. Most noticeably it tends to confuse spells cleanse (1) and exhaust (3) with heal (7) when they're paired with flash. Using accuracy the model seems highly effective at predicting a heal and flash combination. However it's precision is only 56%, showing that accuracy is misleading and it actually just tends to default to heal when in doubt between the three. From domain knowledge, this is most likely due to heal being considered interchangeable with exhaust and cleanse in certain circumstances.

Strangely enough it was able to predict 'Other' with a relatively high accuracy of 70%, meaning the model could distinguish between popular and niche combinations.

The size of the decision tree is largely due to there being 157 champions in the dataset, however the decision making process is quite simple. It splits first by role, then by minion_kills, and the rest is solely based on champion choice.

Evaluation

Although the model was relatively successful, it is important to acknowledge the limitations of our methods and the dataset. The large majority of the data was inconsequential to, and ignored in, the final model. Additionally, two of the three features that we did use - role and minions_killed - were simplified, potentially restricting their utility.

None of the features we binned were selected in the final model. This could be attributed to the method we used to bin or the decision to bin in the first place. The model may have performed better if we had not binned these features, despite the fact that it would decrease model fitting time efficiency.

We suspect that the inclusion of additional data features in the dataset could increase the supervised ML model's predictive capability. For example, the opponent of the player has a significant impact on spell choice and would assist the model. Length of match would also allow us to normalise the data points against time, which may improve NMI scores.

To conclude, the supervised learning model was able to use some features in the dataset to predict the player selected summoner spells with a reasonable accuracy. However, there are changes that can be made to both our methods and the dataset itself to improve on the model.

References

1. [The Dataset]Andrew Suter, “LoL Challenger Soloq Data (Jan, Kr-Na-Euw).” Kaggle, 2022, doi: 10.34740/KAGGLE/DSV/3193532. https://www.kaggle.com/datasets/andrewasuter/lol-challenger-soloq-data-jan-krnaeuw
   Note: The data has been modified to suit the assessment criteria.