A2 Report Group 149

Authors: Charlie Davis-Tope , Jayaram Annadurai , Himmat Singh Sandha, Ider Byambadorj

Contents: 1

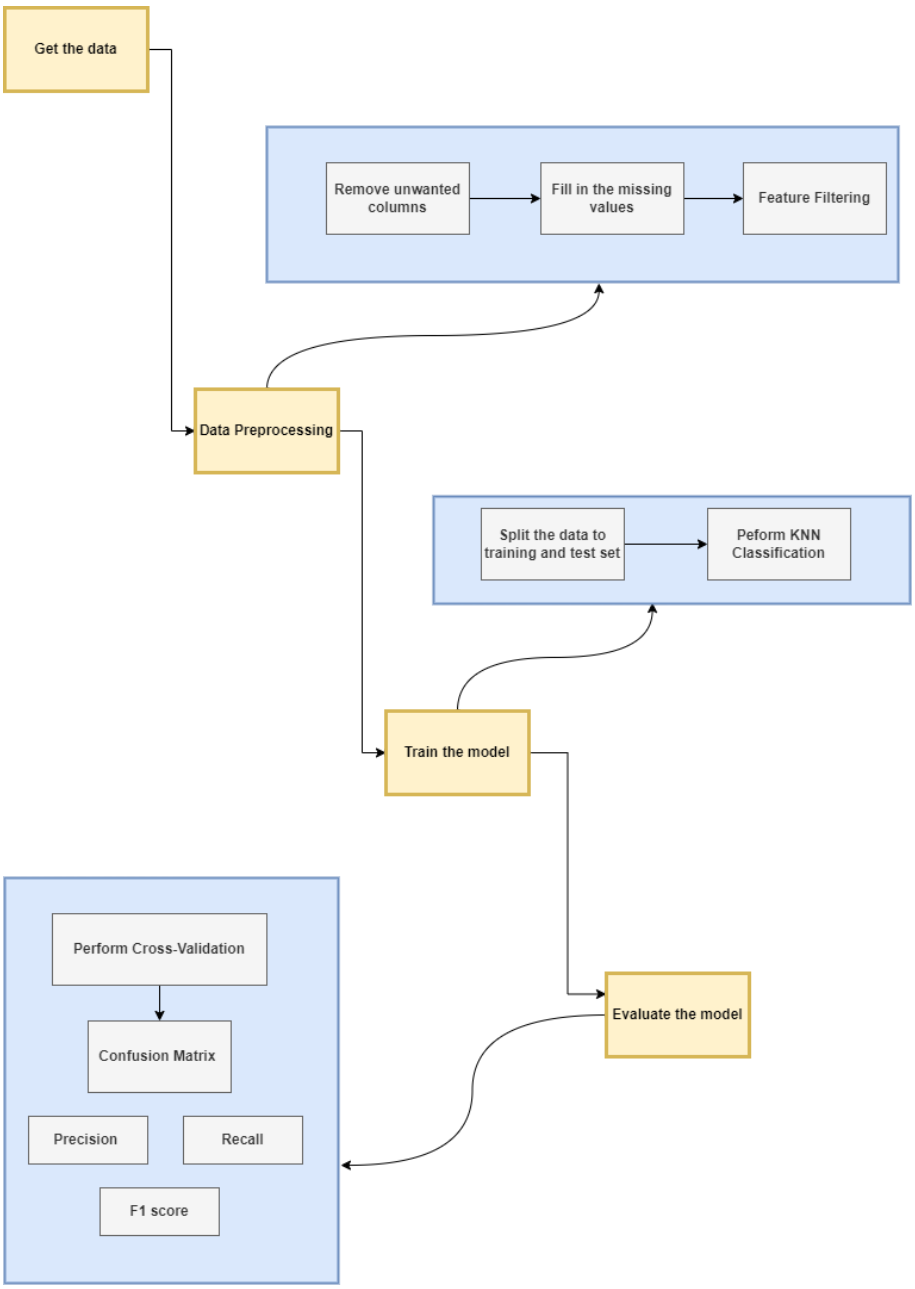
(Image via Riot Games)

Aim

The aim of the project is to determine a predictive model for the ‘role’ of a player using their gameplay data in League of Legends (LOL). The results may intrigue game developers attempting to improve players’ gameplay experiences through using such a predictive model. The model could be used to implement a matchmaking feature that matches other players with different roles to the same team to avoid overlapping roles. Improvements to the game based on the model would be beneficial to all players.

The role evaluation could help new players decide a role to play by suggesting them a role similar to how they’re already playing the game, removing some complexity for a beginner to start. The hypothesis is that the role of a player can be predicted given the player’s gameplay performance.

Our plan for achieving our aim is displayed in the flowchart:



Dataset

Three datasets, based on region (EU: European, KR: Korean, NA: North American), contain statistics of individual player performance, randomly selected from games from the month of January across three regions, from the game LOL.

Each instance in each dataset is randomly chosen player's gameplay performance stats from a unique game and contains the columns:

- d_spell: times d_spell used
- f_spell: times f_spell used
- champion: specific character chosen
- side: player's team(red/blue)
- assists: number of damaged opposing teams players that died without the player dealing the final hit

- damage_objectives: damage to objectives
- damage_building: damage done to the building structures (includes, damage_turrets: damage done to turrets)
- deaths: times health is reduced to zero
- kda: kill-death-assist ratio
- kills: number of damaged opposing team players who through player hitting causes them to die
- level: amount of levels a player has.
- time_cc: time crowd controlling opposing players using special champion abilities
- damage_taken: damage received from all sources
- turret_kills: number of turrets who through the player hitting them causes them to be destroyed
- vision_score: amount of vision provided by a player using “wards”
- damage_total: total damage given by the player to any source. gold_earned: total gold earned
- role: categorises the main actions the player does to support the team. (ToplaneJungle/Other)
- minions_killed: amount of “creeps” killed by the player (few/many)

A major issue with the data quality is that most entries contain NaN / missing values. The entries remaining for EU, KR and NA were 2621, 2555 and 2595 respectively after removing entries with NaN values whereas the total number of entries for EU, KR and NA were 5771, 5697 and 5760 respectively.

With the exception of damage_total, gold_earned and minions_killed all other attributes contained missing values.

Pre-processing and Wrangling

Feature Selection

Each region's dataset would be processed individually to take into account differences in playstyle across regions which may determine how a role is played. A ‘region’ feature for the combined dataset could have been added and attempted to be processed but because the region contained 3 categories, a more advanced ordinal encoder would need to be used which the group lacked the understanding to implement. No data linkage was required.

The data needed to be cleaned so that machine learning techniques could be applied properly. Feature selection was used to select only necessary features of the dataset that were relevant for determining the role feature and to remove ones that created noise improving efficiency. The improved efficiency in the model is essential in order to expand the model for larger datasets, like all players of League of Legends instead of the top players.

The champion feature was dropped as it contained too many categorical variables. The champion feature could have been processed but this would lead to a massive increase in each dataset’s size The sparse matrix, adding 147, 155, 157 new columns for KR, NA and EU datasets respectively would be expensive to run and the possibility of new added champions to the game would require the model to be continuously updated.

The other categorical data could be processed with a binary encoder.

Using domain knowledge, the ‘side’ feature could potentially be dropped due to side intuitively not having any impact on the role of a player. However the side given influence what colour a player sees their character as which could influence the data as seen in a given study, so further testing was required.

The ‘d_spell’ and the ‘f_spell’ features were both dropped because both features had very high variance as well as there was no easy way to fill in missing data (NaN value) that contributed to a meaningful understanding of the data.

Correlation

The correlation between all the remaining numerical features that had not been dropped were plotted in a heat map visualisation for each region to give a rough idea how features were correlated (Figures 1, 2 and 3) From these heatmaps, several clear very high correlations were clear from the data so further analysis was investigated. Values that have a **Pearson's correlation** higher than 0.8 were determined to be significantly correlated with each other. In the EU and KR datasets, it can be seen that "gold_earned" is highly correlated with "damage_total" and "level". To increase efficiency, we decided to reduce the number of attributes that had similar features. As such, we decided to drop “damage_total”, “level” features and keep the “gold_earned” feature. The same conclusions were made in the NA datasets but the extra feature,”turret_kills” , was also highly correlated and also dropped.

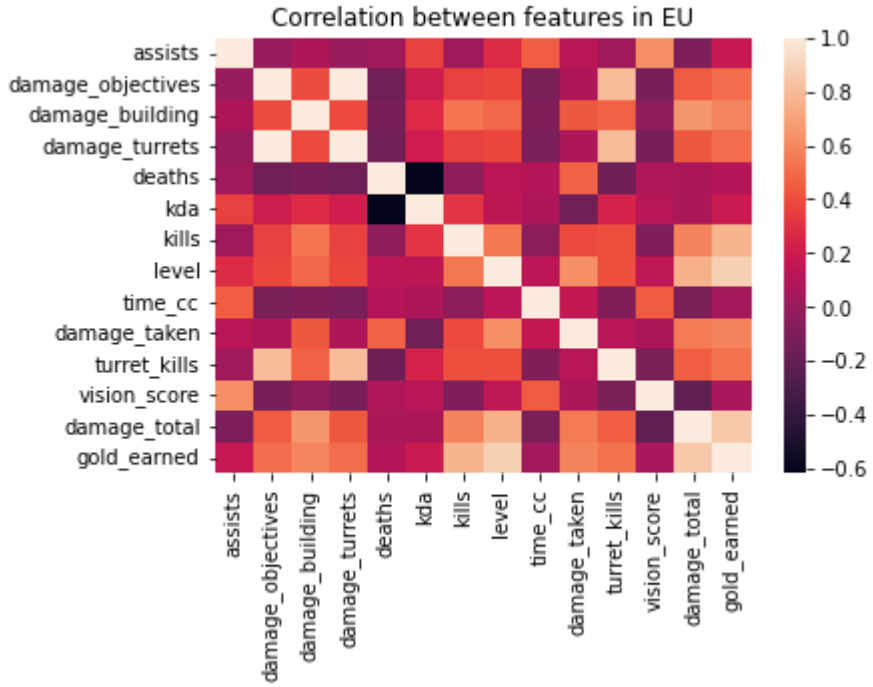


Figure 1. Heatmap of Correlation between features in EU dataset

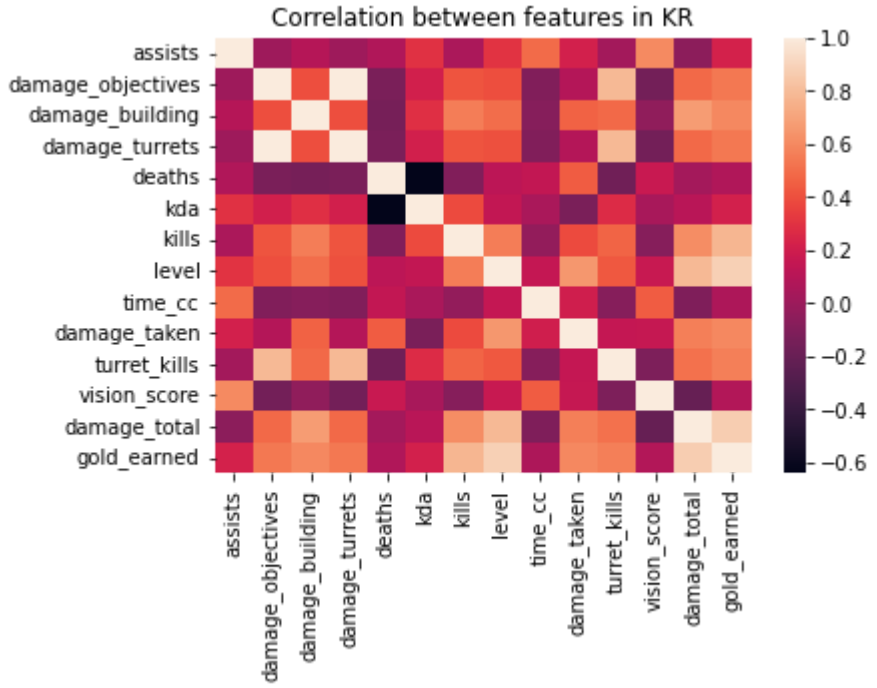


Figure 2. Heatmap of Correlation between features in KR dataset

Figure 3. Heatmap of Correlation between features in NA dataset

Filling in missing values

The data was then divided into categorical data and numerical data to sort out NaN values.

Since most attributes in the numerical data had left-skewed distribution (Fig 4,5,6), we used median values to fill in the missing values. In this project, we used **SimpleImputer** from

sklearn to fill in the missing values. For the categorical data, we removed all entries that had missing values as there weren't any meaningful ways to fill in missing values.

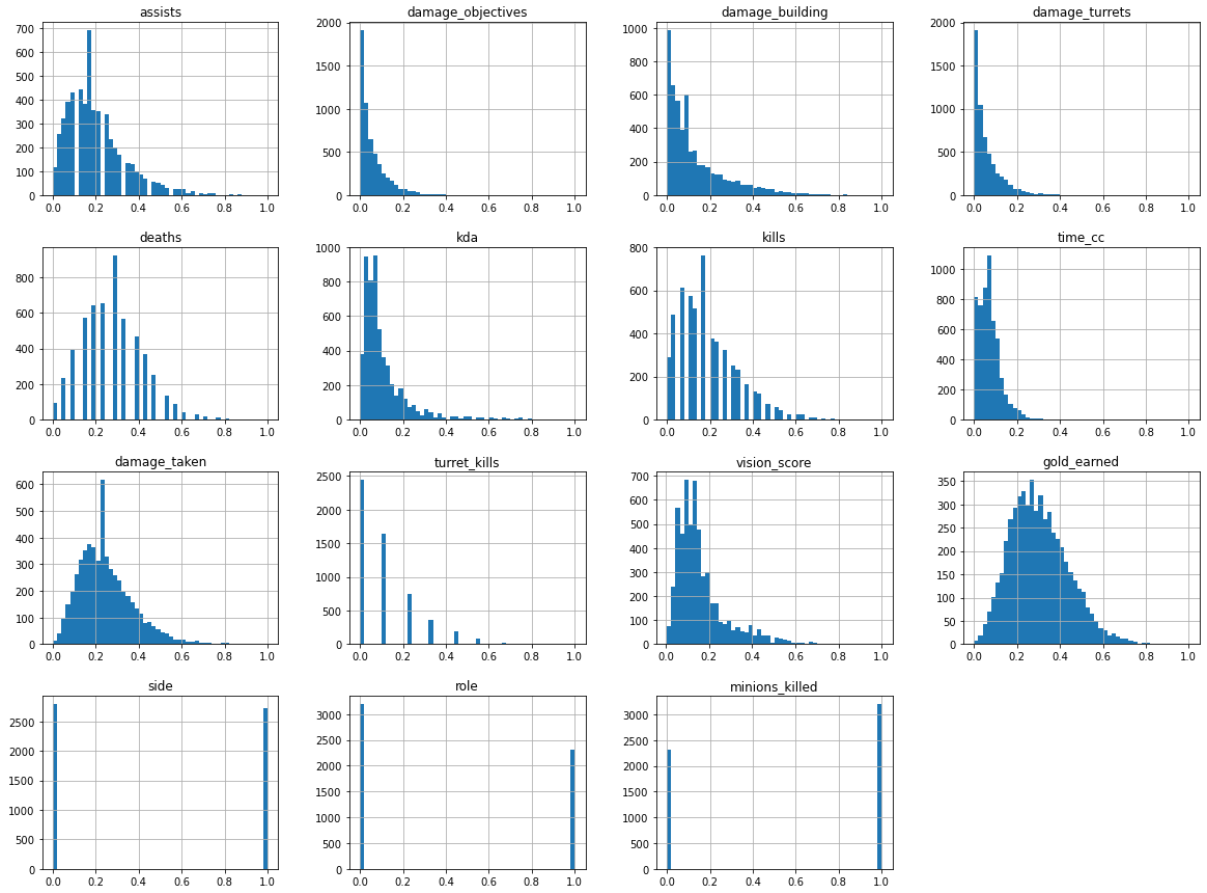


Figure 4. Histograms of distributions of features in EU dataset

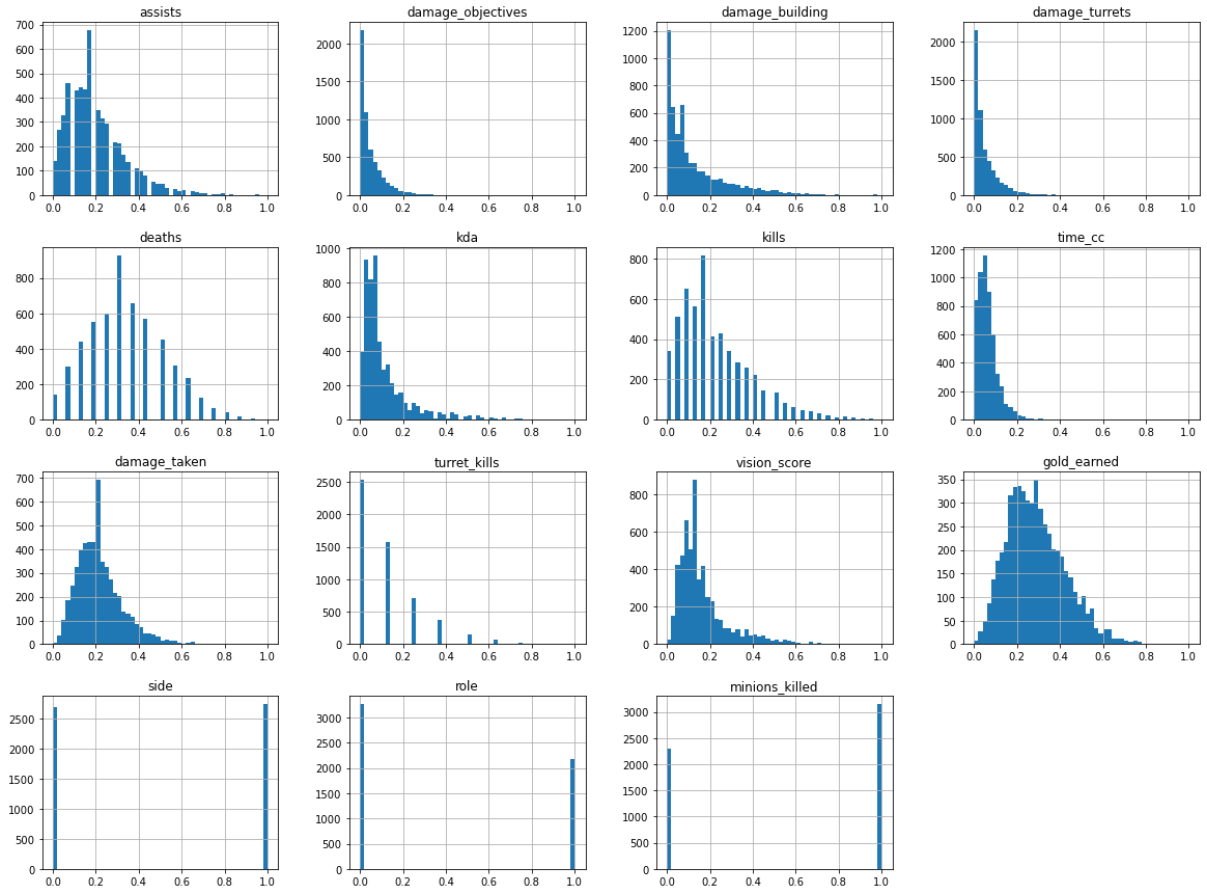


Figure 5. Histograms of distributions of features in KR dataset

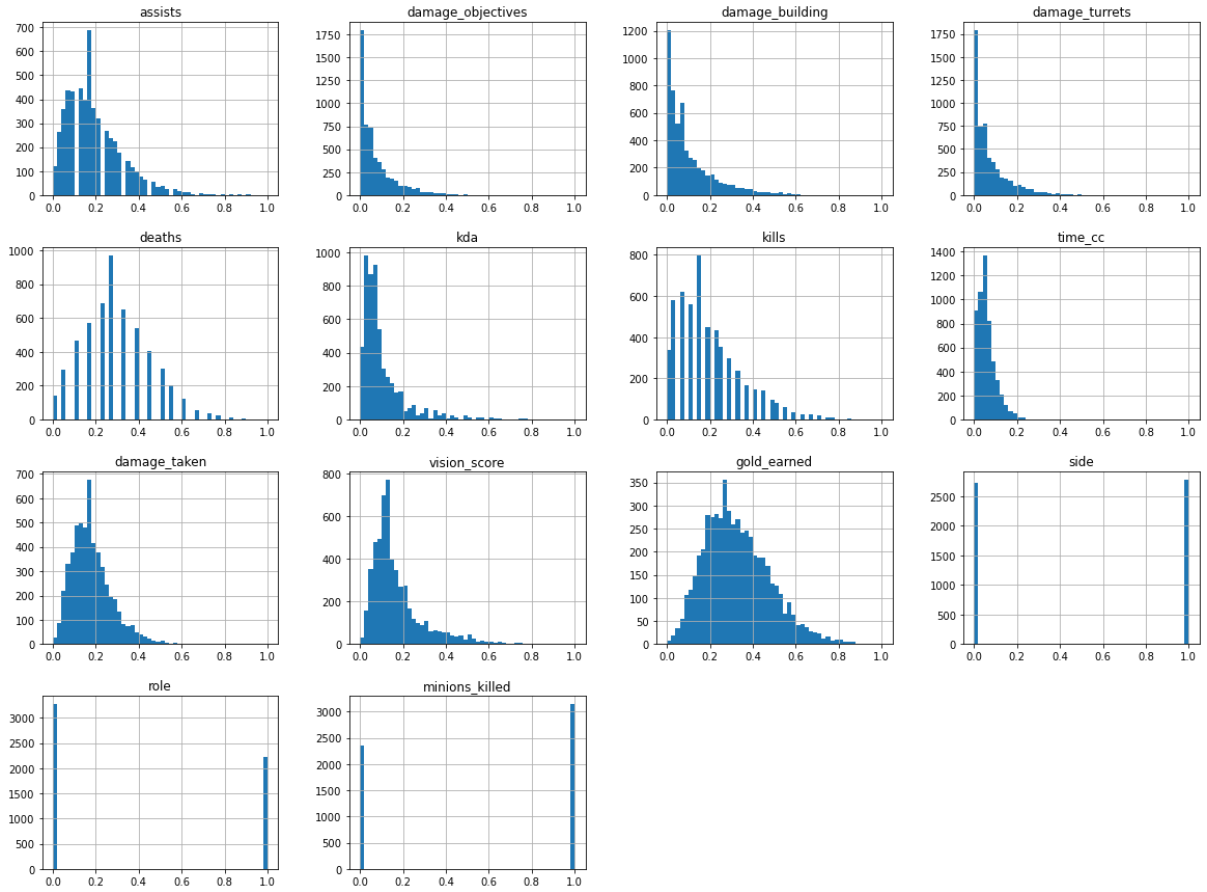


Figure 6. Histograms of distributions of features in NA dataset

Preparing the data

Feature scaling: The data was normalized as there is huge variation in the means for different attributes which would cause some attributes to dominate more in the classification algorithm providing inaccurate predictions. Therefore, we used the **MinMaxscaler** for numerical datas to normalize the entries between 0 and 1. This is done by subtracting a value from a minimum value and dividing with a difference of maximum value and minimum value.

Feature Filtering: Each categorical data feature only had 2 options. Ordinal encoding could be used to turn the categories into “1”s and “0”s. The **Chi test** was used to determine which data between “side” and “minions_killed” were more important. According to the result, we decided to drop the “side” column and keep the “minions_killed” attribute. (Fig 4, Fig 5, Fig 6)

All steps were then implemented on all three datasets.

Analysis Methods

Methods

In alignment with the research question, we used classification methods. As our data had many attributes, the decision tree was not suitable for the dataset. Thus, we chose to use the K-nearest Neighbours algorithm.

Training-Test Split

Random split and **Stratified Shuffle Split** were both tested to find the best option and it was found that the random split had much more error. This error was likely due to baise in the selected data.

As shown below, after testing both random split and stratified shuffle split methods we were able to identify that stratified shuffle split had less % of errors as compared to random split (0.02% to 1.61% respectively rounded off to nearest 2 decimal places) (Figure 7).

	Overall	Stratified	Random	Random %error	Stratified %error
0.0	0.591075	0.591185	0.600608	1.612824	0.018696
1.0	0.408925	0.408815	0.399392	-2.331234	-0.027024

Figure 7. Evaluation of Random Split and Stratified Shuffle Split

Preliminary Analysis

To understand how each attribute is related to each other using a combined data set, we calculated the correlation among all numeric variables as part of the feature selection process (Figures 1, 2 and 3). We plotted each variable's distribution using a histogram (Figures 4, 5 and 6). Histogram of distributions of features before normalization is included in the appendix. We found out the general relationship between variables but no clear correlation between variable 'role' and other variables were shown. This arose the need for our supervising method to answer the research question.

Modelling

K-NN classification

We used the k-Nearest Neighbours algorithm as our supervising method to answer the research question. To determine the appropriate number of neighbors to consider in the KNN nearest neighbours model, we performed several tests with different k-values on all three datasets. We then found that the F1 scores were the highest when 5, 3 and 3-nearest neighbours models were used on EU, KR and NA datasets respectively. Thus, we determine the k values of the k-NN algorithm to be 5, 3 and 3 (Tables 1, 2 and 3).

Cross Validation

To evaluate the model accurately, we used K-fold Cross Validation. Also to perform the sampling in a way which doesn't introduce bias, we selected **StartifiedKFold** split based cross-validation.We performed several trials to determine the appropriate value for K and found that average accuracy was the highest when 10-fold, 6-fold and 5-fold cross validation were performed on EU, KR and NA datasets respectively (Table 4). The average accuracies were 0.819526, 0.809552, and 0.815883 on EU, KR and NA datasets respectively. This demonstrates that the model is consistent with different datasets and different splits.

Discussion

The results show that a predictive model based on gameplay stats can predict the role of a player to a degree of success. According to the result, we found that the role can be

predicted with accuracy of 0.8330, 0.8174 and 0.8025 across regions Europe, Korean and North American respectively. The fitting of our model revealed that F1 scores of 0.7904, 0.7605 and 0.7491 were seen across three regions (Table 5). Despite the results seeming to support our hypothesis the player number of league of legends must be taken into account for successful application. Confusion matrices are included in the appendix.

	Europe (EU)	**Korean (KR)**	**North American (NA)**
Accuracy	0.833031	0.817431	0.802548
Recall	0.747845	0.723112	0.728090
Precision	0.838164	0.802030	0.771429
F1	0.790433	0.760529	0.749133

Table 5. Final Performance metrics for three regions (Rounded to 6 decimal places)

Assuming an accuracy of 85% of the model, for correctly predicted results for the 180,000,000 average monthly players in 2022, still leaves 27,000,000 players who will have their role misinterpreted which isn't appropriate for the given applications.

Although the difference is small, we found that all measures (accuracy, recall, precision and F1) of the model performed on the EU dataset was higher than that performed on KR and NA datasets. This may be caused by the difference in playstyles. Further research should be done to give a reasonable explanation.

Evaluation

Limitations:

As explained in the dataset section, the given dataset contains too many missing values. As k-NN is a sensitive algorithm, a small tweak in the dataset could lead to different results. Although we've handled the missing values in the preprocessing and wrangling section, the bias coming from the data should not be ignored.

With data only coming from the month of January, the current model may quickly become outdated with new game updates and player strategies. These changes possibly affect gameplay stats for roles that this model would not predict well.

Furthermore, since the dataset contained challenger players' gameplay data, our model may not work well on different levels of players' data. The model should be tested on different datasets to answer this question.

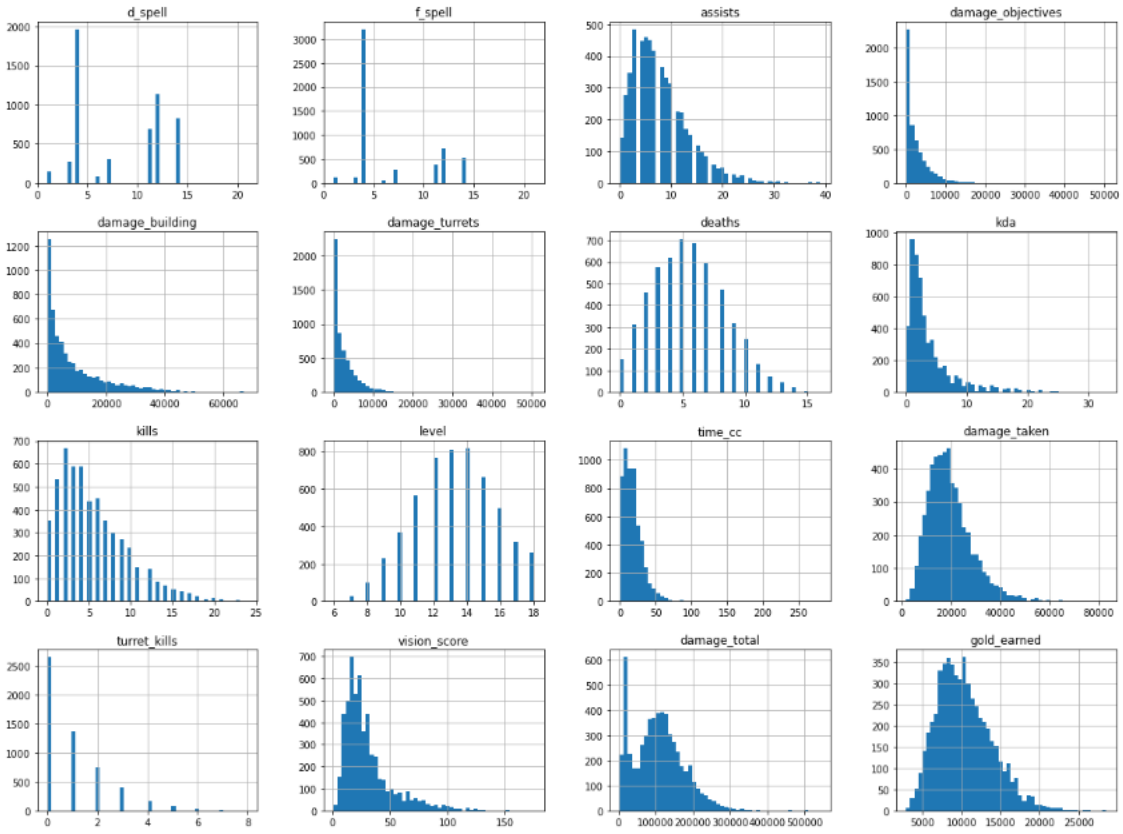
Future improvements:

The machine learning model could be expanded to be able to predict more specific roles as opposed to just 'Other' and "TopLane_Jungle". Through obtaining more data, by surveying players and game developers asking what type of roles exist, more complex variations could be tested through supervised learning to split up the "Other" into more roles. "TopLane_Jungle" can be split into smaller sub-roles of "TopLane" and "Jungle" and even further sub-roles i.e "aggressive top-lane" and "top-lane and stealer".

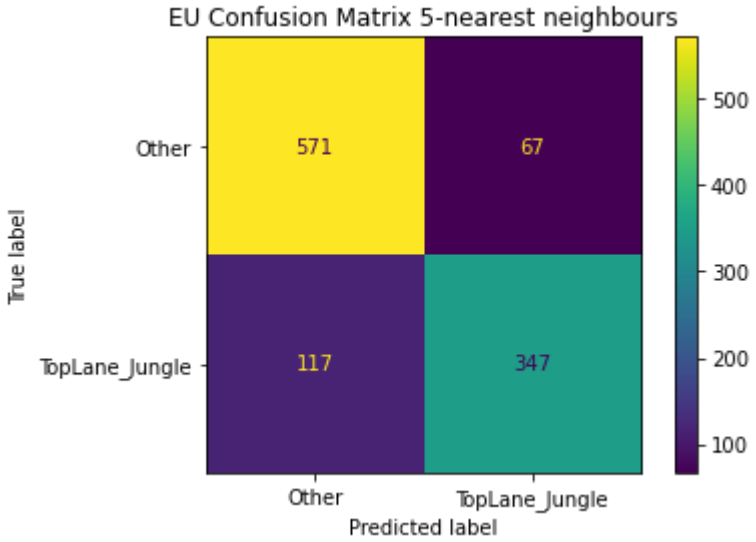
A recommender system could be built on top of the predictive model using the number of games as certain roles and the champions they play to recommend certain champions for people who like to play specific roles. LOL has a weekly roster of 16 characters which are free to play. An extra bonus spot could be added as the “personal role recommendation” to people who played a minimum amount of time in a given week. The bonus element will likely reduce community frustration.

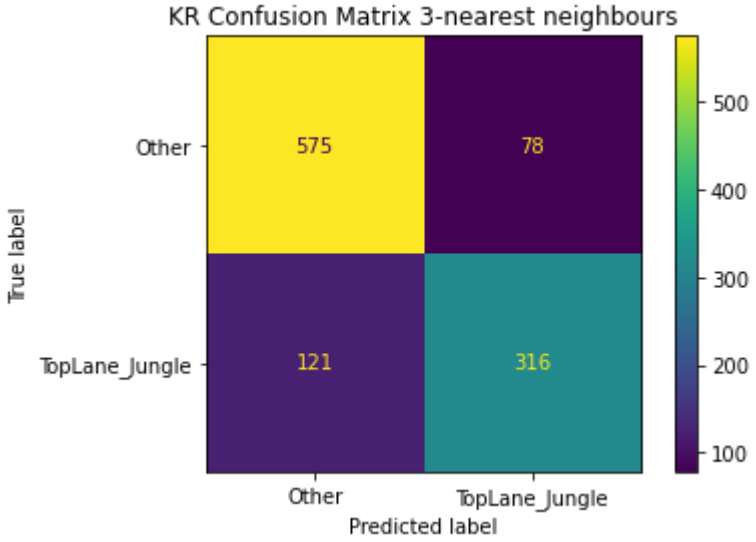
Appendix

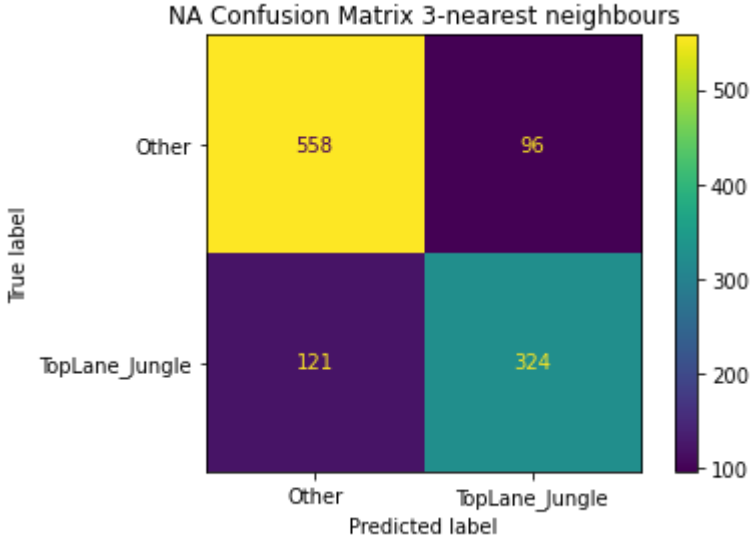
Histogram of distribution of features before normalization



Confusion Matrices of k-Nearest Neighbours model


EU Confusion Matrix 5-nearest neighbours
Other
TopLane_Jungle
True label
571
67
117
347
Other
TopLane_Jungle
Predicted label
500
400
300
200
100


KR Confusion Matrix 3-nearest neighbours
Other
TopLane_Jungle
True label
575
78
121
316
Other
TopLane_Jungle
Predicted label
500
400
300
200
100


NA Confusion Matrix 3-nearest neighbours
Other
TopLane_Jungle
True label
558
96
121
324
Other
TopLane_Jungle
Predicted label
500
400
300
200
100

	3-nearest neighbours	**4-nearest neighbours**	**5-nearest neighbours**	**6-nearest neighbours**
Accuracy	0.814882	0.822142	0.835753	0.833031
Recall	0.743534	0.665948	0.758621	0.698276
Precision	0.802326	0.882857	0.836105	0.880435
F1	0.771812	0.759214	**0.795480**	0.778846

Table 1. Performance metrics for different k-values of k-nearest neighbours on EU dataset (Rounded to 6 decimal places)

	3-nearest neighbours	**4-nearest neighbours**	**5-nearest neighbours**	**6-nearest neighbours**
Accuracy	0.816514	0.809174	0.818349	0.819266
Recall	0.713959	0.620137	0.700229	0.638444
Precision	0.806202	0.865815	0.820375	0.877358
F1	**0.757282**	0.722667	0.755556	0.739073

Table 2. Performance metrics for different k-values of k-nearest neighbours on KR dataset (Rounded to 6 decimal places)

	3-nearest neighbours	**4-nearest neighbours**	**5-nearest neighbours**	**6-nearest neighbours**
Accuracy	0.798908	0.803458	0.805278	0.811647
Recall	0.716854	0.624719	0.692135	0.642697
Precision	0.770531	0.850153	0.800000	0.856287
F1	**0.742724**	0.720207	0.742169	0.734275

Table 3. Performance metrics for different k-values of k-nearest neighbours on NA dataset (Rounded to 6 decimal places)

Average Accuracy of k-fold Cross Validation	European dataset (EU) (10-fold)	Korean dataset (KR) (6-fold)	North American dataset (NA) (5-fold)
5-fold	0.810216	0.807027	**0.815883**

6-fold	0.815442	**0.809552**	0.812919
7-fold	0.814081	0.807947	0.814510
8-fold	0.819064	0.805197	0.814288
9-fold	0.816802	0.805877	0.815655
10-fold	**0.819526**	0.807951	0.816329

Table 4. Average accuracy of k-fold Cross Validation of the model among three regions (Rounded to 6 decimal places)

References

Andrew Suter. (2022). LoL Challenger Soloq Data (Jan, Kr-Na-Euw) [Data set]. Kaggle. https://doi.org/10.34740/KAGGLE/DSV/3193532

Spezzy, (April 5 2022), *League Feed: How Many People Play League of Legends? – League of Legends Player Count in 2022 (April)*, https://leaguefeed.net/did-you-know-total-league-of-legends-player-count-updated/

Fandom (2022), *League of Legends Wiki: Free Champion Rotation,* https://leagueoflegends.fandom.com/wiki/Free_champion_rotation

Dreiskaemper, Dennis & Strauss, Bernd & Hagemann, Norbert & Büsch, Dirk. (2013). *Influence of Red Jersey Color on Physical Parameters in Combat Sports. Journal of sport & exercise psychology,* https://www.researchgate.net/publication/235605104_Influence_of_Red_Jersey_Color_on_Physical_Parameters_in_Combat_Sports#:~:text=Hill%20and%20Barton%20%282005%29%20showed%20that%20fighters%20in,fighters%27%20physical%20parameters%20of%20strength%20and%20heart%20rate.

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011, https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011, https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html