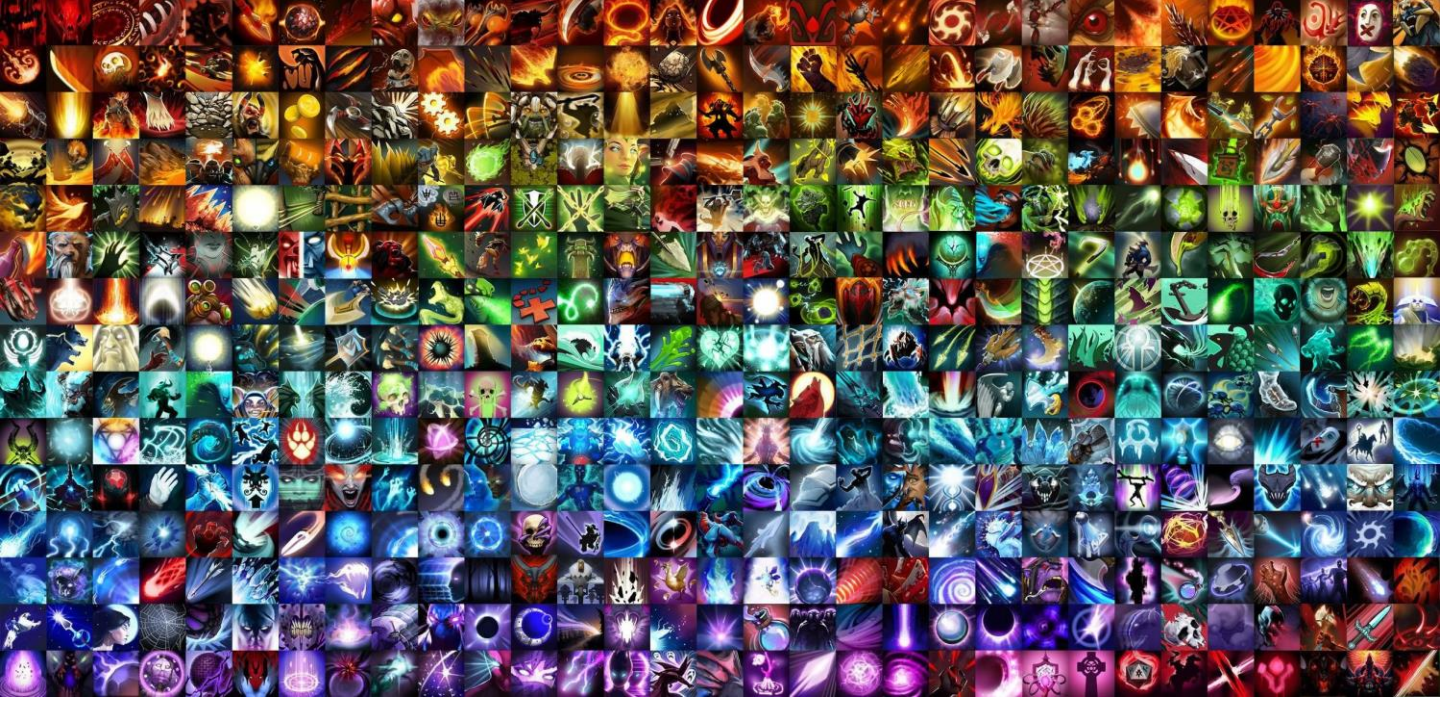
DOTA 2
MATCH WINNER
PREDICTION MODEL

Salvador Robles

January 2021



INTRODUCTION

Why?

Dota 2 has become the most lucrative e-sport in history. In 2021, its premier annual tournament “The International”, which is broadcasted to millions of viewers will distribute a prize pool of more than $40 million to the 5 members of the winning team.

Dota 2 steep learning curve and overall complexity has made it subject of famous machine learning experiments, such as the “Open AI Five”.

Commentators and viewers struggle to interpret teams’ composition as well as what team has better chances of winning.

Audience

Individual players, viewers, streamers and commentators of tournaments would all benefit from having a winning team predictor.

Having a clear initial picture, before the match starts, of what team has better chances of winning as well as better understanding each team’s composition, would help define better strategies and comment or anticipate what strategies to expect from both teams.

Data Source

For the data, we choose a Kaggle data set downloaded from the API OpenDota, which includes in-game and end-game information of 500k players within 50k matches.

This is a sufficient size dataset to develop a good predictor model

DATA

Original Dataset

Files downloaded included mainly of *in-game* and *end-game* information of players and the map.

None were ready to be used and several transformations were done. In summary, we worked with the following files:

1. **Heroes**
   - hero's names and ids.
2. **Players**
   - end-game players' metrics, showing each player's accrued information regarding n° of kills made, n° of times dead, gold earned, etc.
3. **players_time**
   - in-game players' metrics (in minutes), showing evolution of information regarding amount of gold earned, experience earned, and last hits performed.
4. **match**
   - outcome of the match, what team won.

The **first task** of this project, to *categorize heroes into groups* required a DataFrame with each hero's name and its end-game metrics. This would allow to create clusters of heroes based on certain metrics (e.g., number of kills, amount of gold earned, …). This DataFrame was generated using information of the "players" dataset grouped by hero and median values of the distributions.

The **second task** of this project, to *predict the winner before the match starts*, required a DataFrame with the heroes picked by each team and their categories (generated in previous task).

- This DataFrame was generated using information of the "players" dataset and our own "Categories Data" DataFrame.

The **third task** of this project, to *predict the winner at minute 15 of the match* required information of each team's performance in terms of advantage/disadvantage of gold per minute and experience per minute.

- This DataFrame was generated using information of "players_time" through some transformations to obtain teams' values.

METHOD

Clustering of heroes

There are 112 different heroes that players can select.

To categorize them, we first calculated the typical "behaviour" of every hero, based on the median end-game metrics of players using them.

Then, we applied **k-means algorithm** to create clusters of heroes on different spectrums, such as their "powerspike", this is, how early or late in the match a hero is more powerful comparatively to other heroes.

Predicting winner

To predict the winner both before the match starts and at minute 15 of the match, we choose the supervised machine learning method of **Random Forest Classifier**.

It works well with categorical and continuous data, without needing to normalize or scale the data. The second model contains minute by minute gap in gold per minute and experience per minute between teams.

Hyperparameter tunning with GridSearchCV was applied to both models to optimize performance.

DATA WRANGLING

Problems with original files

The problems with the original files were to reduce the amount of available information to only the necessary for this project. Many files were excluded, and many columns dropped within the files selected.

Problems in creating hero clusters

What's the typical hero behavior? Players may play the same hero in different ways. It was necessary to define a criteria before clustering on what is each hero's "typical" behavior.

> Solution: Typical hero behavior was inferred by grouping heroes on the median end-game metrics, as a proxy of this idea, avoiding using the mean, to avoid distortion.

Some hero spectrums require relative metrics: To understand some of the heroes' spectrums, we had to create our own measures of end-game metrics, which were not available.

> Solution: We created some metrics such as the source of gold by percentage, to better understand the weight on sources of gold, experience, etc.

Problems with in-game information

In-game information, this is, minute by minute **evolution of each team's metrics** (gold earned, experience earned) **was not available.**

> Solution: We calculated each team's gold, experience, gold per minute, experience per minute and the gap on these metrics between teams.

Problems with machine learning format

To predict the winning team before the match starts, we had to format the data of heroes picked in binary, but their **categories could not be input the same way, as it would have resulted in hundreds of additional columns** (each team has 5 players and each player has 3 spectrums: function (6 options), farm dependency (3 options) and powerspike (3 options), meaning 5*2*3*12 = 1200 additional columns)

> Solution: We created DataFrames for team radiant and team dire with one column per hero name (112 cols per team, binary), and then the added 12 classes per team (24 columns) including teams' class hero count (discrete). This reduced additional columns from 1200 to 24 columns.

To predict the winning team at minute 15 of the match we had to include **the gap of gold per minute and experience per minute** between teams at different minutes. This information **was in rows** as a series, and we needed it in columns.

> Solution: We transposed the rows to columns and added them to the model as dependent variables.

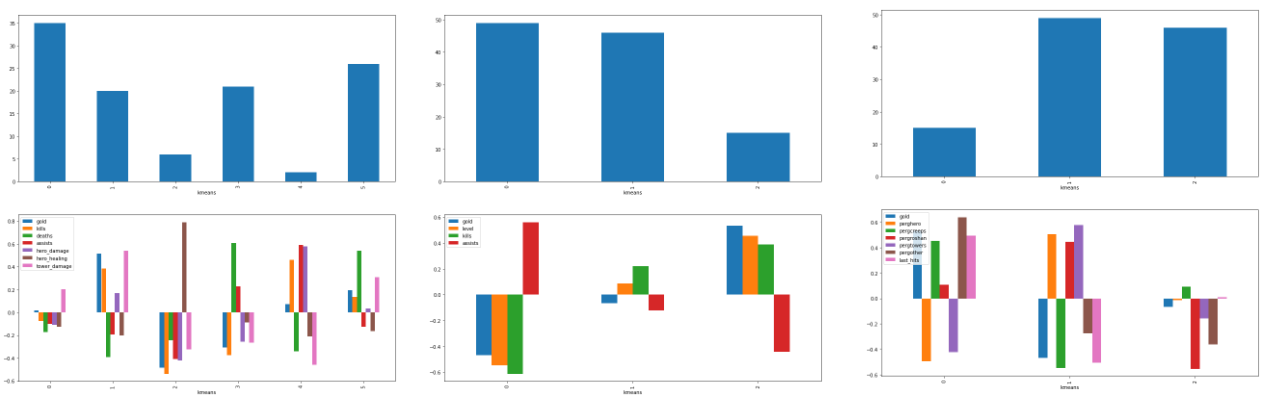
EDA – Hero Clusters

Can we organize the pool of heroes into clusters, aiming at defining a better team strategy?

There are 112 heroes, each with their own abilities, strengths and weaknesses (as in football with strikers, midfielders, defenders and keepers). Every hero requires a different amount of time and resources to become impactful and plays a different role in the match.

Considering that each match duration is different and resources in the map are limited, team members struggle to understand topics such as how to split resources, what to expect from their teammates in terms of gameplay style, or when in the match their heroes will be relatively stronger than the enemy team.

To assess these topics, we worked on the idea of creating "spectrums" of heroes, based on the role they play, the number of resources they need to develop and the moment of the match they peak.



We created the following "spectrums" or categories of heroes:

- **Spectrum of function**: Showing the typical hero behavior. A total of 6 functions were identified.
  - Core Roles:
    - Killers
    - Pushers
    - Killer/Pusher/Tank
    - Killer/Pusher/Weak
  - Support Roles:
    - Healer/Tank
    - Killer/Weak
- **Spectrum of resources**: Showing how much resources the hero needs to become impactful. A total of 3 groups were identified.
  - High Farm
  - Moderate Farm
  - Low Farm
- **Spectrum of powerspike**: Shows the moment of the match where the hero is relatively more impactful than other heroes. A total of 3 groups were identified.
  - Early Game
  - Mid Game
  - Late Game

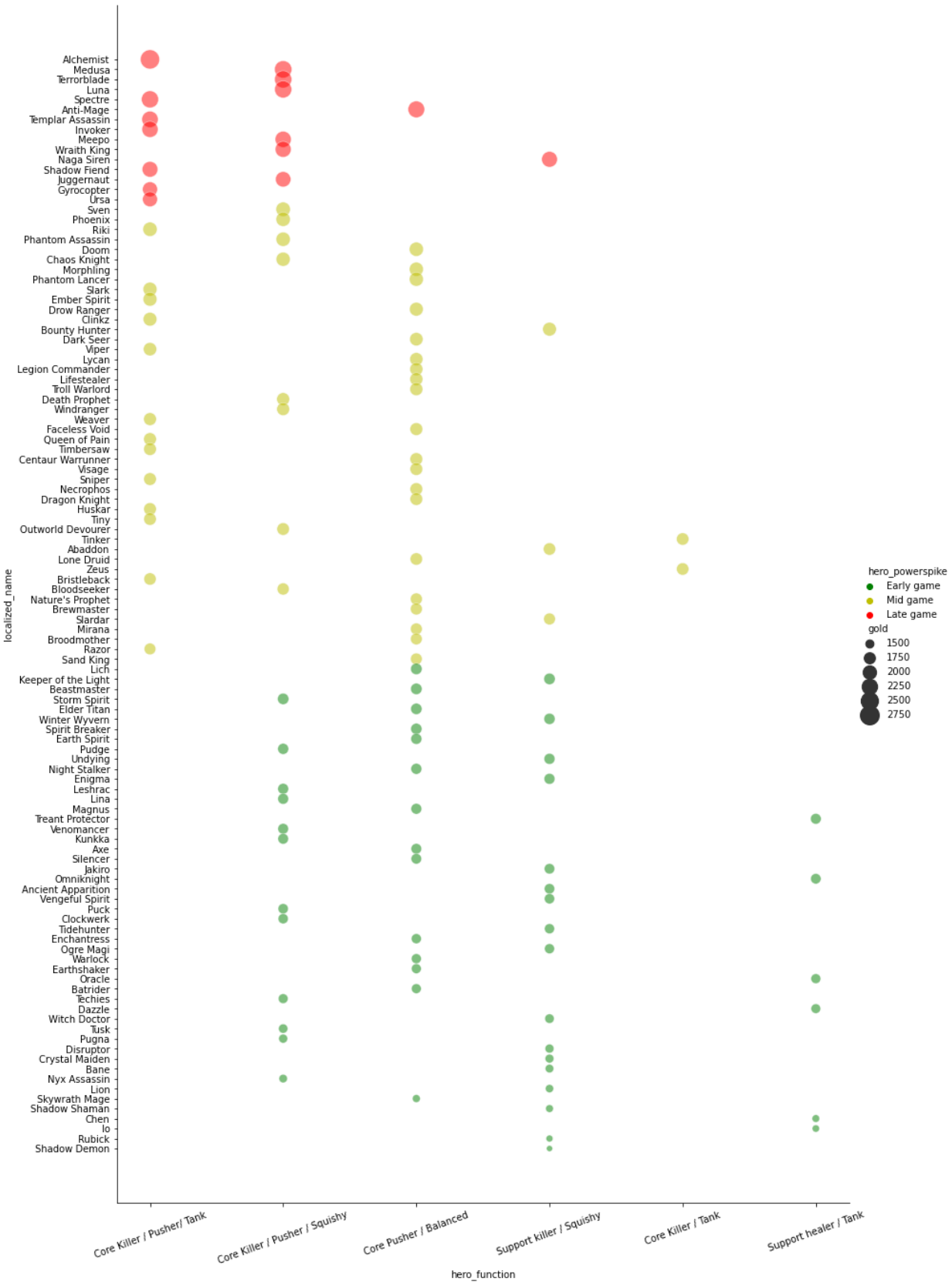
EDA – Hero Clusters

The figure below shows visually every Dota 2 hero according to our defined spectrums.

Hero Function: Knowing the hero function helps to understand if it should focus mainly on their own development (Core Subgroup) or in supporting their Core team mates development (Support Subgroup), as well as what gamestyle better suits each hero.

Hero Powerspike: Knowing the team's powerspikes helps to understand if the best strategy should be going for an early win or a late win.

Hero Farm: Hero Farm is not shown in the figure explicitly but can be deduced from the amount of gold earned by each hero (the size of the dot). Knowing each hero resource requirements to become impactful allows team members to better distribute the limited resources.



EDA – Win Metrics

What metrics allow us to predict what team is going to win the match?

We identified that the main metrics that show what team is winning is the gap between teams in **gold per minute (gpm)** and **experience per minute (xpm)**. These metrics show the amount of gold and experience earned by a team divided by the minute of the match. The team that has an advantage in gpm/xpm is likely to win the match.

Figure on the right shows the gap evolution of gpm and xpm between teams in a single match. At zero mark (red line) both teams are equal. In the positive range, team radiant has an advantage and, in the negative range team dire has an advantage.

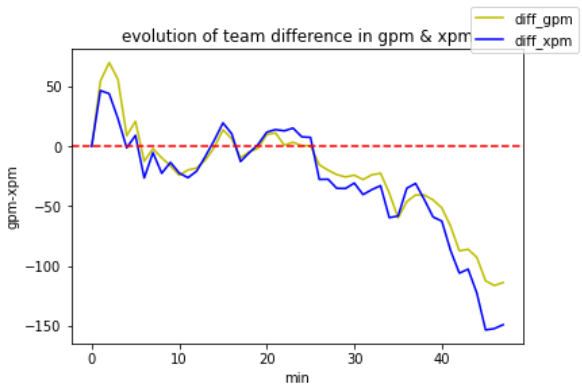
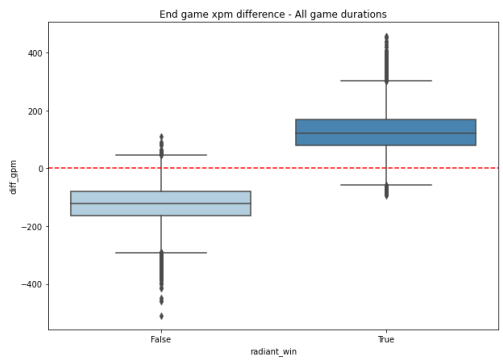
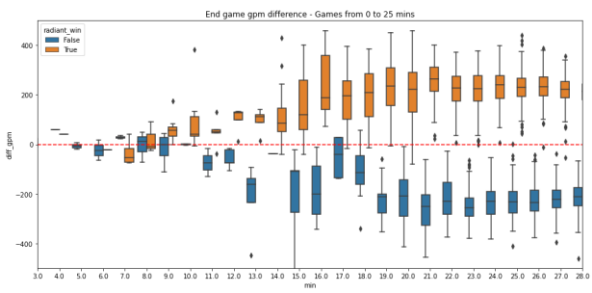
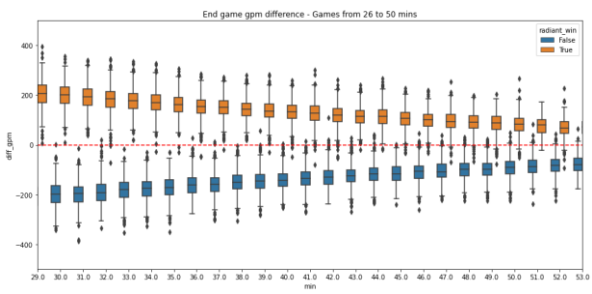


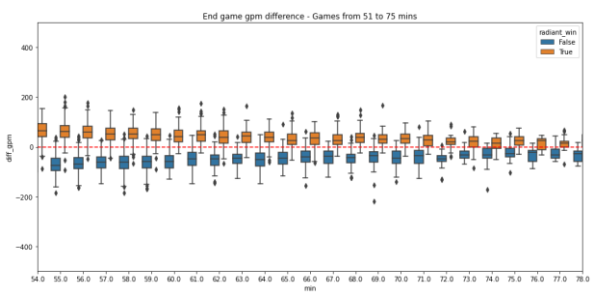
Figure on the right shows the gap in gpm/xpm between winners and losers at all match durations. It can be observed how team radiant (dark blue) won most of the matches when it had a gpm advantage.



To verify that this observation was true in **every match duration**, we decided to observe the gap of gpm between teams in short, medium and long match durations.







In shorter matches (top left) the gap is higher, and the as the match duration increases (mid left), the gap starts to shrink. In longer matches (bottom left) the differences continue to decrease, converging to the zero line (both teams are equal).

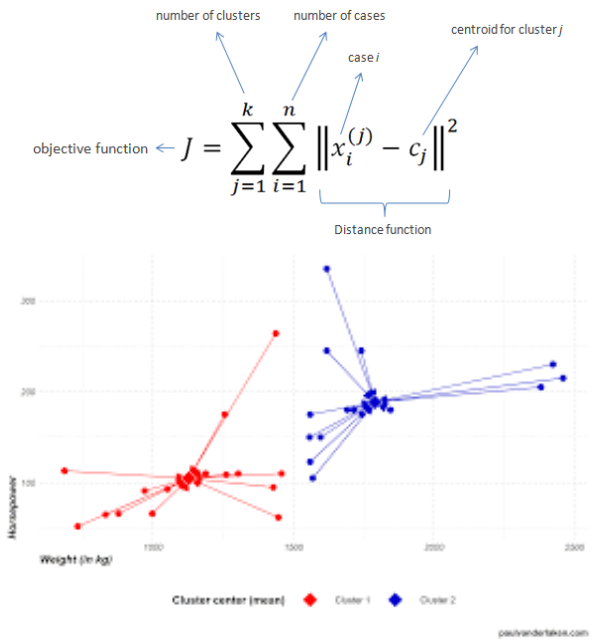
Regardless of this trend the gap exists at every match duration and is significant in the most common match durations (40-45 minutes).

Therefore, **this metric is valid to state what team is currently winning and to predict what team will likely win the match.**

ALGORITMS & ML

For the clustering task of generating the heroes' spectrums we used the k-means algorithm.

The k-means algorithm scales well with large datasets (for future project upgrades) and guarantees convergence of classes. To apply k-means, it is necessary to define the number of clusters "k" to generate. This was important for us as we wanted to keep control of the number of classes to generate in some cases (for example, we choose a suboptimal "k" for the hero functions, to allow a wider variety of hero roles).



One comment to be made is that k-means has trouble with clusters of varying sizes and density. To solve this, however, it is possible to generalize k-means.

Also, centroids can be dragged by outliers and a cluster containing only the outlier may be created. Therefore, it is advised to remove outliers before clustering, in our case this was not a problem.

For the match prediction task, we decided to use Random Forest Classifier and hyperparameter tunning (GridSearchCV)

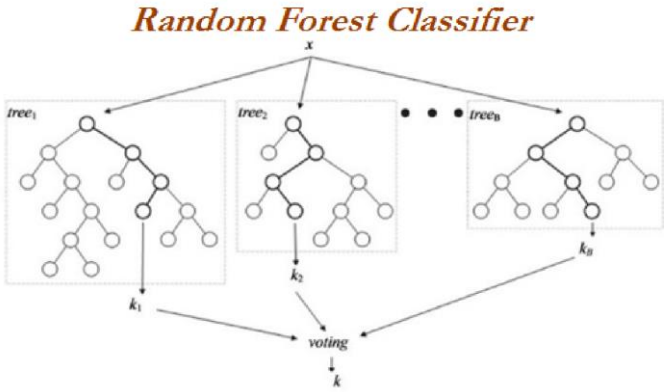
Random Forest Classifier works well in classification problems to the extent that it reduces overfitting in decision trees, helping to improve accuracy, and is capable to handle both categorical and continuous data without the need to normalize or scale the data. As our models contained binary and discrete variables, we wanted to allow it to handle both data types properly.

As drawbacks, it requires much computational power to build trees and combine their outputs.

Also, the model training takes much time due to the combination of the tress and, due to the ensemble of decision trees, interpretability is hard.

$$K_k^{cc}(\mathbf{x}, \mathbf{z}) = \sum_{k_1, \ldots, k_d, \sum_{j=1}^{d} k_j = k} \frac{k!}{k_1! \cdots k_d!} \left(\frac{1}{d}\right)^k \prod_{j=1}^{d} \mathbf{1}_{\lceil 2^{k_j} x_j \rceil = \lceil 2^{k_j} z_j \rceil},$$

$$\text{for all } \mathbf{x}, \mathbf{z} \in [0,1]^d.$$

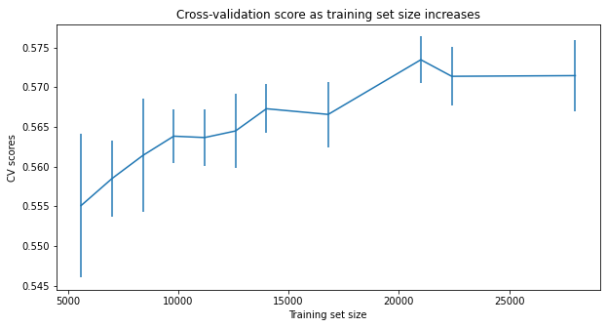


ACCURACY

A player could predict right the winning team 51% of the times if they always predicted the same team to win.

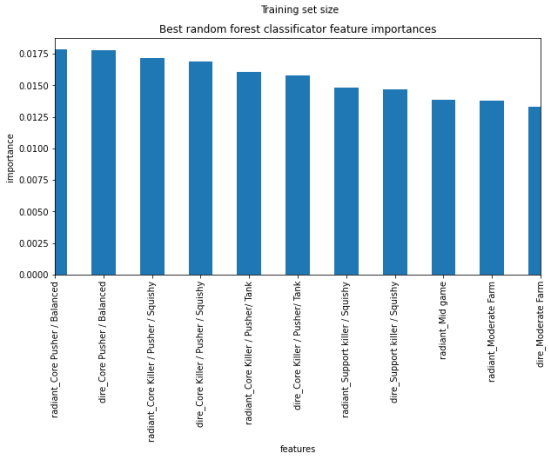
Predicting winning team before the match starts

Before the match starts, only using information regarding the players heroes selected and their classes, **the accuracy is improved to around 58%.**



This improvement is significantly higher than the baseline case considering the amount of uncertainty in the scenario.

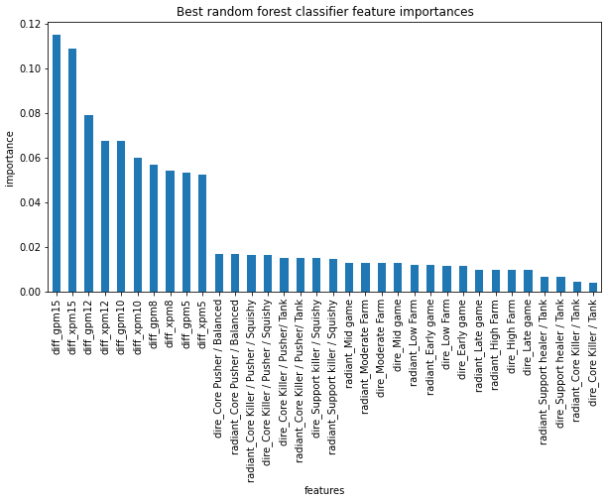
Most important features for the model are the hero classes created through the k-means clusters.



In relation with data quantity assessment, it seems that there is enough data, as the CV score plateaus at some point.

Predicting winning team at minute 15 of the match

At minute 15 of the match, using the gap in gpm and xpm between teams on minutes 5, 8, 10, 12 and 15, together with the classes of heroes selected by players, **the accuracy improved to around 70%**



This improvement is significantly higher than the baseline case and is still working with much uncertainty, as most matches last around 40-45 minutes.

The most important features for the model, unsurprisingly, are the gpm and xpm gap at different minute marks.

In relation with data quantity assessment, it seems that there is enough data, as they CV score slowly increases with more data, but not significantly.

NEXT STEPS

- Deployment of the model through a web-based platform so it is available for users

FUTURE IMPROVEMENTS

- Increase the number of “spectrums” of heroes to allow more classes to be created within such spectrums
- Introduce to the model items purchased and abilities upgraded by hero
- Estimate match duration based on initial draft and/or some in-game metric
- Predict the skill level of individual players and add this information to the model
- Update the database with Dota 2 matches of 2021