Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Recommender Models for items in the game 'Destiny 2'

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Jorge Jiménez García

Tutor: Rajesh Jaiswal, Eva Onaindia de la Rivahererra

2023/2024

TU Dublin, Tallaght Campus

BSc Project

Recommender models for items in the game 'Destiny 2'

Jorge Jiménez García

Department of Computing

Supervised by

Dr. Rajesh Jasiwal
Department of Computing

30 April 2023

Declaration

I hereby certify that the material, which I now submit for assessment on the programmes of study leading to the award of Bachelor of Science, is entirely my own work and has not been taken from the work of others except to the extent that such work has been cited and acknowledged within the text of my own work. No portion of the work contained in this thesis has been submitted in support of an application for another degree or qualification to this or any other institution.

Jorge Jiménez García
30 April 2023

Acknowledgements

To my supervisor Rajesh Jasiwal for what amounts to infinite patience. To family, friends and everyone who is, was and will always there.

9th of August, 2019:
4, 3, 2, 1...
Don't jump, don't jump!

List of Figures

List of Tables

Contents

Abstract

Currently, players of the popular video-game '*Destiny 2*' waste an excessive amount of time skimming through their items trying to figure out if they are worth using or not. To fill this unexplored niche, and with the goal of assisting them and reducing the time spent, models for the area of popularity prediction and item recommendation are explored. An analysis of the problems of data for this particular domain are laid out. A novel approach of using a heuristic metric is developed for scoring and assessing performance for this problem context. Hyperparameter tuning is carried out for all proposed models. Models are proposed and then tested on data incoming from future, unseen game releases, obtaining good results when predicting items in future releases. Potential biases present in the model are discussed, and a Model Card is provided. Future work is laid out.

1 Introduction

1.1 The problem domain

In the popular video game ‘Destiny 2’, available for all modern gaming platforms, players construct loadouts of 3 weapons to participate in different multiplayer activities, with a multitude of weapons and items with which to duel each other in the player-vs-player activities (PvP), or as it is referred to in-game, The Crucible, where players engage in casual 6v6 team-based combat.

Users obtain these items from a variety of in-game activities with ranging amounts of difficulty, and depending on the activity, with up to five other players. Items appear with randomized attributes, and players spend countless hours repeating activities, or ’grinding’, to obtain their perfect copy of items.

However, the vast range of possibilities on them, be it their stats or possible attributes makes players often overwhelmed with choice when picking a good loadout. This burden is especially acute when new sets of items are released into the game. Whereas for current items the playerbase has already figured out which items are good or not, for newly released additions, that ’figuring out’ process can take a very long time. An evidence of this is Reddit’s *r/sharditkeepit* subforum[1], where players all over the world ask others about how good their items are or if they are at all worth keeping. The subforum counts close to 63k players amongst its ranks.

In the game, items are generally randomized. This means they come with some modifiers to the base attributes of the item, which are generally not that important since most weapons come with the same modifiers available to the. Importantly, they come with 2 random traits that generally alter the behaviour of the item.

As an example, pictured below in Figure 1 a personal copy of the *‘Long Shadow’* sniper rifle. For the Player Versus Player (or PvP) environments of interest, this would generally be agreed to be a bad version of the weapon.

These traits, or perks as they are commonly referred to, are what players generally look at to see if a weapon is good or not. An example of the available traits to the

[1]`https://reddit.com/r/sharditkeepit`

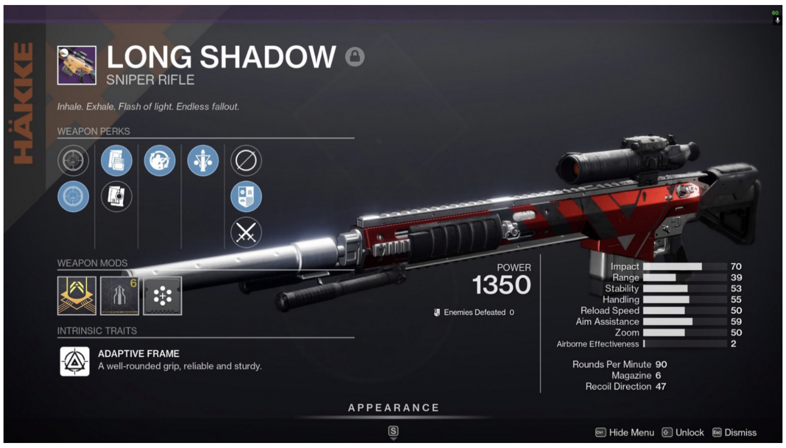


Figure 1: A personal copy of the *'Long Shadow'* item, with the random traits *'Field Prep'* and *'Triple Tap'*

'Long Shadow' which has a fair amount of available traits are shown below in Fig. 2 marked in red and blue respectively. The other columns represent stat modifiers mentioned previously and are generally the same among all weapon types.

1.2 Research question

Several community websites track weapon 'popularity', however they are ranked by their performance, meaning a weapon that tends to be more effective at defeating opposing players is deemed as 'more popular' than others. That stat can often be misleading, since it does not really measure popularity, but effectiveness. In an effort to learn of ways to know which items are actually popular, even if they are less effective, the question of why an item is actually popular naturally arises. In turn, trying to predict how popular a weapon is is a clear next step. This formed the following research question:

Given an item's properties, can it be determined how popular it is?.

1.3 Methodology

To tackle this, there will first be a discussion on how to collect the data necessary to undertake this project, as well as an analysis of general caracteristics of the collected data and some relevant particularities in Section 2. With information

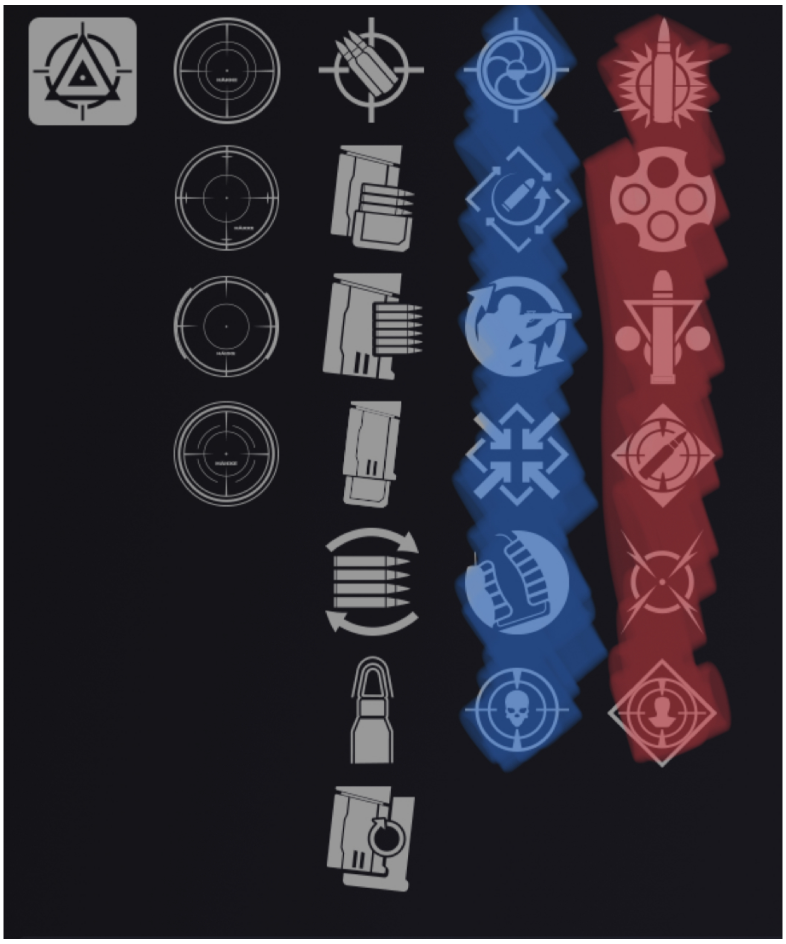


Figure 2: *'Long Shadow'* perk selection

about the data in hand, literature review and research will be performed on how to solve potential problems, read up on previous or similar attempts at this kind of problem, as well as model types which have shown promising results in Section 3. These models will then be tuned, developed and evaluated in order to judge which of them is best at solving this particular problem as part of Sections 4 and 5. With a model chosen, potential biases present in the process will be discussed and a model card elaborated. To conclude, a summary of the development and future lines of work will be outlined in the closing Section 10.

2 Collection and discussion on Data

2.1 Data Collection

2.1.1 Collection Algorithm

The dataset is manually gathered using the available set of public API services the game developers provide[2].

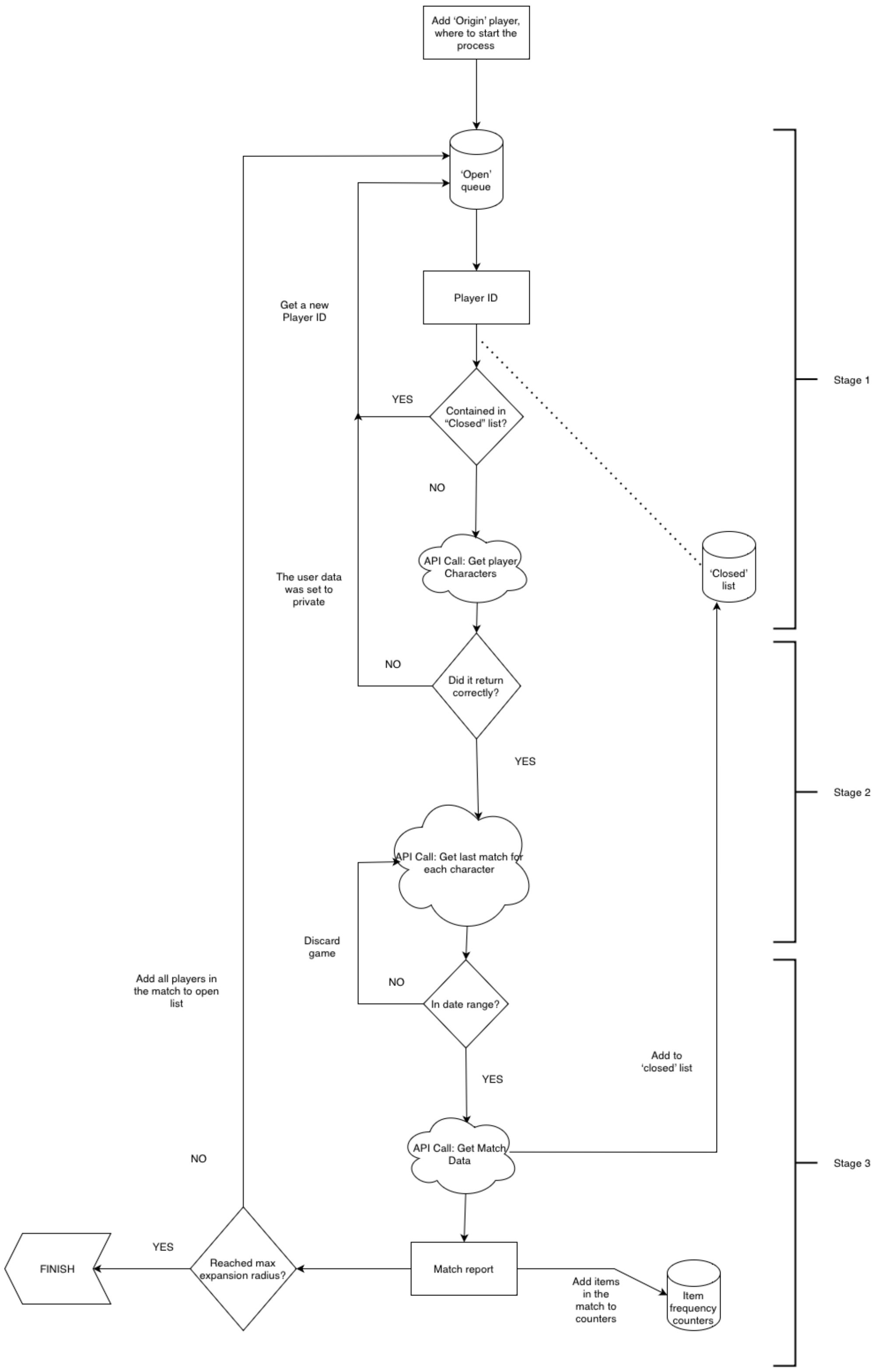
The strategy is based on a snowballing search, illustrated in Figure 3. To develop this gathering script, Python was chosen due to its strong set of pre-existing libraries as well as ease of development. In particular, the '*json*', '*requests*' and '*pickle*' modules were a big draw for manipulating API responses, performing queries and storing gathering results respectively, as well as integrating with all common ML frameworks.

The search process is divided into three stages: 'Stage 1' is dedicated to getting new, unseen player IDs to query. To do this, IDs found from previous iterations or the origin list are used and compared with IDs in the 'closed' player ID set, which have already been processed. If they have not been processed yet (meaning they are not present in the 'closed' set), the API is queried for the player's information and it is passed to the next step. Note that if a user has their account set to private, their information is discarded and repeat with a fresh player ID.

'Stage 2' gets a set of matches from the user. It uses the player information obtained from the previous stage to get their in-game characters. Players can have up to three of them, and each of their character's data is queried for their last match played. Since the search can be restricted in time, once the games are retrieved, they are checked to see if they are within the time interval of interest. If they are, they are passed on to Stage 3. Otherwise, they are discarded.

'Stage 3' is dedicated to gathering the metrics to form the dataset. To this end, the match's data is inspected and all weapons used throughout the match are logged. With this information, counters are updated for each unique weapon and their associated ID. Once complete, the match ID is added to its respective 'closed' list.

[2] `https://bungie-net.github.io/multi/index.html`



Add 'Origin' player, where to start the process
'Open' queue
Player ID
Get a new Player ID
YES
Contained in "Closed" list?
NO
API Call: Get player Characters
The user data was set to private
'Closed' list
Stage 1
NO
Did it return correctly?
YES
Stage 2
API Call: Get last match for each character
Discard game
NO
In date range?
Add all players in the match to open list
YES
Add to 'closed' list
API Call: Get Match Data
NO
Stage 3
YES
FINISH
Reached max expansion radius?
Match report
Add items in the match to counters
Item frequency counters

Figure 3: Search strategy diagram

This is also the last step of an iteration. If after processing all matches in the queue the maximum depth has been reached, indicated with the ‘radius’ parameter, the search is stopped. If not, all players found in the match are added to the ‘open‘ queue and the search is repeated from Stage 1.

Following the stages shown in the diagram, the algorithm works by making use of an ’open’ queue and a ’closed’ set, where players to be processed are stored, and those who have already been used for data, respectively.

2.1.2 Personally Identifiable Information

To comment on privacy concerns of this search strategy, no personally identifiable information is recorded or accessed at any moment in time. When obtaining player information, no emails, names or phone numbers are contained in the returned information, as the API design does not allow it. Only tangential information linked to a person is accessed (Username and Platform), but even so it is never stored in the dataset and is only used for logging purposes. Items IDs are then used to query game databases to extract their in-game stats and attributes.

2.2 Data Structure and Pre-processing

The dataset consists of 470 item instances consisting of 172 features.

Items present attributes describing their archetype, sub-archetype and ammunition type. Additionally, they present attributes representing their capability in battle, for instance how stable or their fire rate, among others. The majority of weapon classes have common attributes, but some, such as Rocket and Grenade launchers have attributes that are only relevant to that particular class, like ’Blast Radius’. In the dataset, these are all represented and filled with the value -1 for attributes that are not applicable or have no value. A description of common attributes is present in Annex Table 1.

More importantly is the way of representing perks, which, as mentioned in the introduction, are expected to be the main drivers in predicting popularity. With the goal of keeping perk data as categorical, these are represented One-Hot-Encoding, resulting in a sparse matrix where the X-axis row represent the perk, and the Y-axis column the item, for the perk data. This unfortunately means our data has very high dimensionality. To mitigate this, a distinction between these perks, that is, if they can appear on the left or right slot of an item, is disregarded and instead they are folded into one, representing if they can appear on any of the two slots.

As for other pre-processing steps of these attributes, normalization is not required, since all values have existing bounds discussed in Annex Table 1). Other attributes, such as the weapon sub-class or archetype is label encoded, since despite the fact that this is a categorical variable, there is a hierarchical relationship among the categories (Mukherjee, Garg and Roy 2023). This hierarchical nature is a result of the overall stats an item may have, which are correlated with certain categorical variables, meaning there is a presence of multicolinearity in our data.

Following up on the study of multicolinear data, a heatmap of correlation can be found in Annex Figure 1. Evaluating multicolinearity by this graph, some relationships can be seen in the upper left corner of the heatmap as expected from the 'Archetype' variable denoting a relation between others such as 'Stability', 'Range' or 'Handling' among others, as mentioned in 1. Next, the column for the Y-axis features 'Swing Speed' and 'Guard Resistance' can be seen to display a few bright correlation points with several of the perks lower down in the X-axis of the matrix. This denotes perks that are only found on Swords, which are the only weapon types to have values different to the null indicator -1 for these numerical features. This also explains the clear correlation 'square' forming alongside the diagonal between these two aforementioned numerical variables.

Back to label encoding, this is the same process and rationale to create the three target classes: 'Popular', 'Niche' and 'Unpopular', parting from the Global Usage Rate (GUR), a percentage representing how many times out of all items seen this item was detected.

As seen in Figure 4, there is a sharp increase in the sum of popularity for the highest 10%, which then grows slower until the 50% mark and goes on to grow very slowly onwards. These changes in slope are used to define the classes, where 'Popular' is assigned to the green coloured section in Figure 4 - items in the top 10% of

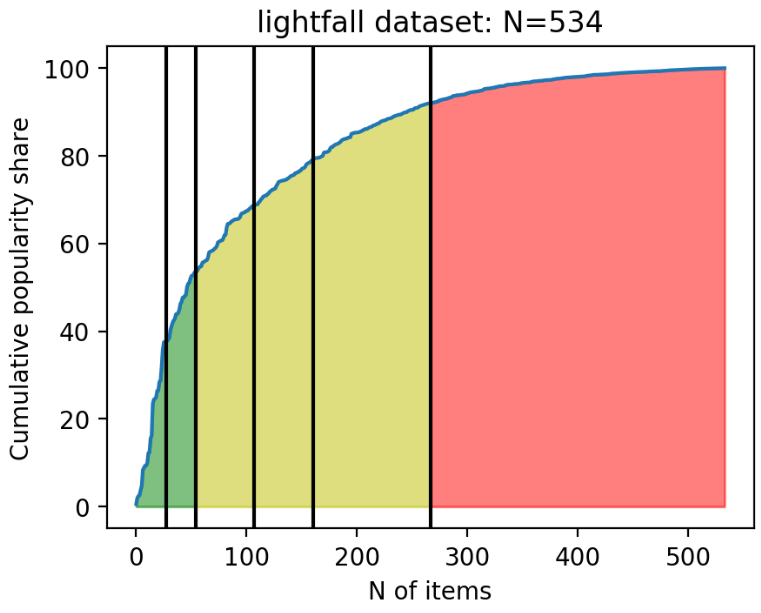


Figure 4: Cumulative Sum of 'Global Usage Rate', 'Lightfall' dataset

popularity, and so on for 'Niche' and 'Unpopular'. Following the aforementioned rationale by Mukherjee, Garg and Roy 2023, this categorical target variable is label encoded to preserve the hierarchy between the classes.

3 Literature Review

3.1 Previous Work

Much work has been done in the area of recommendation, but specifically for video-games, research seems to be scarce, with the majority of it focused on recommending games themselves instead of using them within the game.

For recommender systems for items, Looi et al. (2019) has a similar goal to this project's for a different game or problem domain. They used Rule-Based systems as well as Logistic Regression to good results. Bertens et al. (2018) focus their recommendations on micro-transactions or sale of items, making use of Extremely-Randomized Trees (ERT) to great results when predicting the next item to be purchased by a player.

For 'Destiny' in particular, Joshi et al. (2019) produce a system collecting and assessing a wide range of factors of teams of players aiming to give recommendations for improvement. This included a section related to item recommendations but it was from the standpoint of modifications a player might do to improve their previous results, not as a general recommendation, and took features outside of the ambit of the item itself to achieve them. Drachen et al. (2016) make use of item usage data for profiling players and categorizing them based on their item usage.

3.2 Candidate Models

On the topic of with model selection, some structure in the data makes it suitable for certain models over others, as well as some expected or desirable traits for our final model.

Firstly, since the main problem of choosing an item is the vast amount of possibilities for one, it would be of interest to gain an understanding of why an item is considered popular. This makes explainability a desired trait of our model. For this aspect, a Decision Tree is proposed on the basis of its inherent explainability (Blanco-Justicia and Domingo-Ferrer 2019).

Furthermore, one would expect that different criteria are used to decide on different kinds of items. This type of decision making process resembles the *Ensemble* family of models, and from these a Random Forest model is chosen, for which some model-specific techniques have been developed to attempt to understand its predictions (Sepiolo and Ligęza 2022).

There has been much work on the topic of popularity prediction. In Trzciński and Rokita 2017, Support Vector-based models were successfully used to predict the popularity of online videos. It is noted however, that the exact type of Support Vector model proposed in the paper is not applicable for the purposes of this project since it is explicitly suited for video-based content. Additionally, models that can easily overfit are of interest thanks to works by Jaderberg et al. (2017) stating that models may benefit from fitting to an entire population rather than a sample in terms of generalization of a model. Despite the fact that this refers to

NN-development, it is still worth assessing for traditional models.

As discussed previously in Section 2.2, our data presents the problem of having low sample size and high dimensionality. Work on data presenting this issue has been done. Liu et al. 2017 show workings on a problem of these characteristics but in general the features-to-samples ratio is not close to the shape of the data used, which prior to pre-processing presents a 3 to 1 ratio. Kosztyán, Kurbucz and Katona (2022) state that "The number of features is more than 10 times the number of samples" as an example of the dimensions of the data matrix. Therefore, NN-based models are expected to perform poorly due to the small sample size compared to the feature dimensions.

3.3 Generation of Synthetic Data

The issue of low sample size of data is hard to overcome. For data with features following a statistical distribution, Fowler et al. 2020 propose the use of Kernel Density Estimations, however since it is known that the distribution of perks follows no probability function, and that it would not be possible to accurately assign a target variable to the generated synthetic sample. This is a general issue for all studied synthetic data techniques: The target variable cannot be assigned to the synthetic sample in a way so that it can be known if it is a correct class for the sample.

4 Model Development and Performance

4.1 Performance Metrics

This being a recommendation system with a class imbalance in the data, accuracy is not the best metric to study. Accuracy for the two classes that generally should not be recommended is irrelevant, as long as they do not get misspredicted as the *'popular'* class. Balanced Accuracy or F_1 score may seem apparently better metrics, but due to the nature of recommendations the accuracy for the *'unpopular'* class is not of interest.

With this in mind, a heuristic-driven approach is developed specifically for this problem context. A previous attempt of using heuristic approach to model scoring and evaluation is Wu, Flach and Ferri 2007.

Weighted Failed Prediction Credit, or 'WFPC' is a heuristic metric based on the principle that the impact misclassifying a *'niche'* as *'popular'* is lesser than misclassifying an *'unpopular'* item, being in general the impact of Type I errors being considered low. The metric is calculated as follows:

$$WFPC = TP_{Popular} + (1/(Area_{Niche} + Area_{Popular})) * FP_{Niche} \tag{1}$$

Where $Area_{Niche}$ and $Area_{Popular}$ are, as per Figure 4 and by definition of the classes, 40% and 10% respectively. This also means that the perfect score for this metric is 0.1, which is the entire area of the *'popular'* class. $TP_{Popular}$ and FP_{Niche} are normalized over the size of the dataset and represent a percentage. For these thresholds, the metric results in:

$$WFPC = TP_{Popular} + (1/2) * FP_{Niche} \tag{2}$$

It is important to note the metric's maximum value for these thresholds would be 0.5, where items with true labels *'popular'* and *'niche'* are classified as *'popular'*. Optimally, the value would be 0.1, representing only the percentage of *'popular'* items in the dataset and the FP_{Niche} term being 0. Due to this, it is important to support this with additional observations, such as confusion matrices.

4.2 Model Parameter tuning

The final set of models consists on a Decision Tree, Random Forest ensemble and Gradient-boosted Decision Tree, a Support Vector Machine Classifier, a Bernoulli Naive-Bayes Classifier and a dense Neural Network. These were tested with different sets of parameters via Grid Search and making use of Stratified 10-fold Cross Validation, due to the fact that some of the target classes are severely underrepresented. Note that not all models were subject to parameter exploration due to the lack of tunable parameters, as is the case with Bernoulli Naive-Bayes, or the lack of need for tuning, such as with Random Forests (Probst and Boulesteix 2017), where a default of 1000 estimators was used.

Grid Search was guided to optimize and pick a best performing model based on the WFPC metric. The 'plunder' dataset was used, and models use the sklearn API (Pedregosa et al. 2011). The set of parameters for each model is present in Tables 1 to 4.

Parameter	**Value**
Max_features	All
Max_depth	20
Min_samples_per_leaf	3

Table 1: Decision Tree tuned hyperparameters

Parameter	**Value**
N_estimators	1000
learning_rate	0.1
Max_features	All
Max_depth	5
Min_samples_per_leaf	2

Table 2: Gradient-boosted Decision Tree tuned hyperparameters

Parameter	**Value**
C	50000
Gamma	Scale

Table 3: Support Vector Classifier tuned hyperparameters

Parameter	**Value**
Hidden_layer_sizes	200, 200
Validation_fraction	33%
Max_iter	50

Table 4: Multi-Layer Perceptron tuned hyperparameters

4.3 Model Performance

Using these tuned parameters for all models, the WFPC metric for each of them follows in Table 5. Note that metrics were rounded to 5 decimal positions.

Model	**10-Fold CV Metric**	**Full Training Metric**
Decision Tree	0.05745	0.07553
Random Forest	0.00747	0.02340
Gradient-boosted Decision Tree	0.03151	0.08830
Support Vector Machine	0.03830	0.09149
Bernoulli Naive-Bayes	0.03404	0.06170
Multi-Layer Perceptron	0.01383	0.01702

Table 5: Performance metrics for all models

Additionally, Confusion Matrices for the fully fitted models are also presented in Figures 5 to 10.

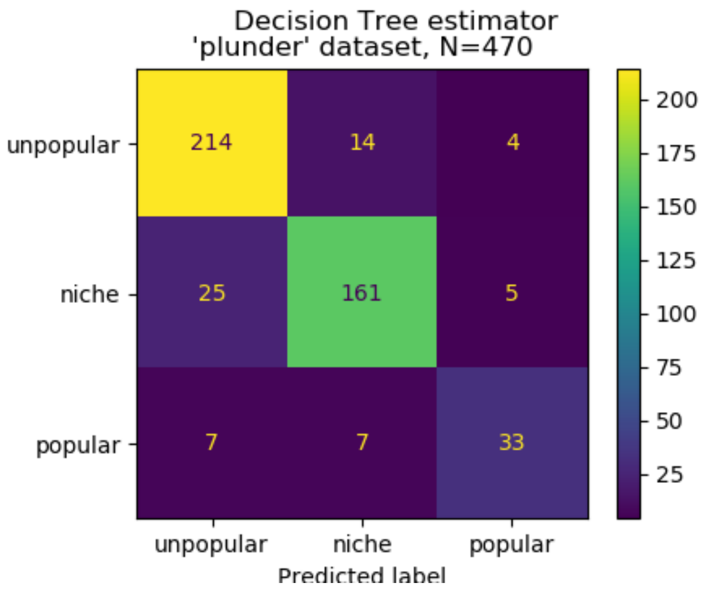


Figure 5: Decision Tree Confusion Matrix

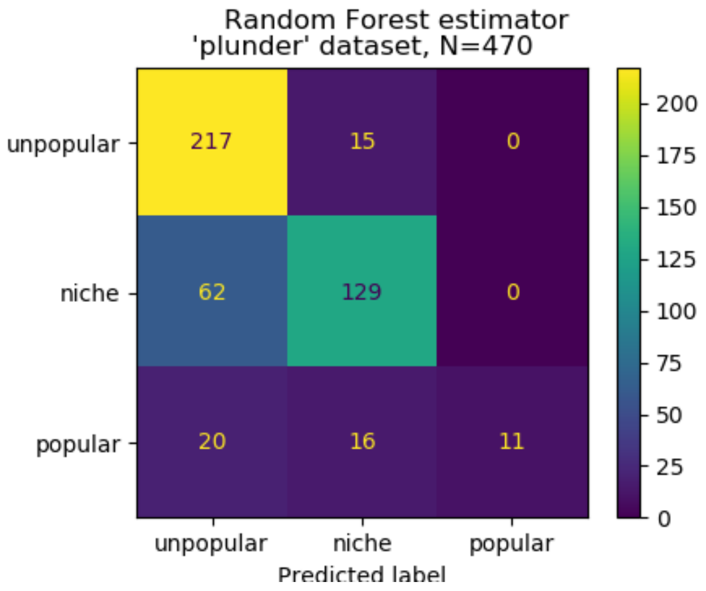


Figure 6: Random Forest Confusion Matrix

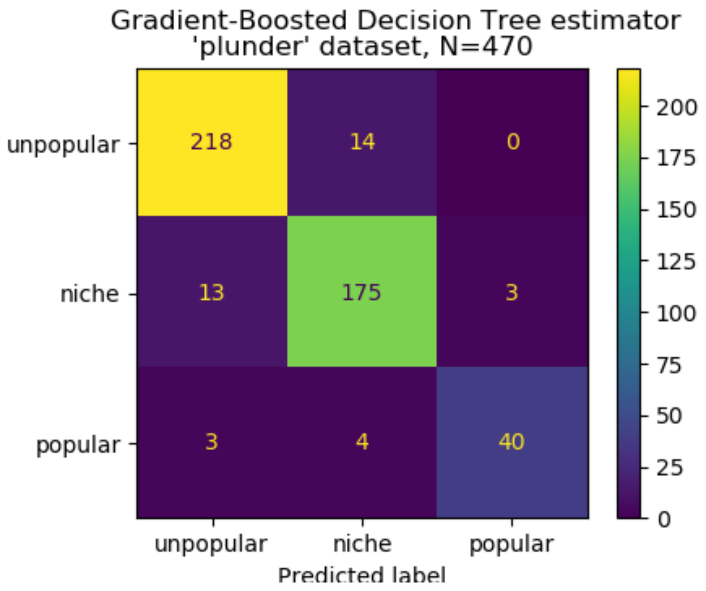


Figure 7: Gradient-boosted Decision Tree Confusion Matrix

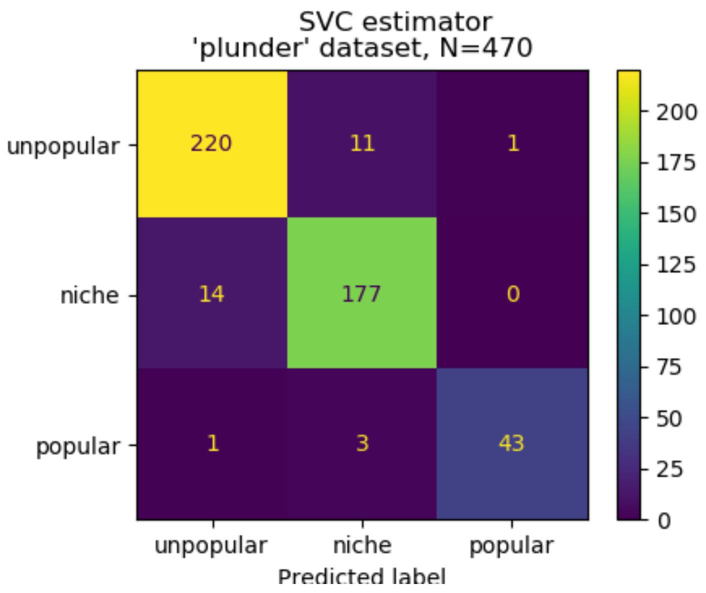


Figure 8: Support Vector Machine Confusion Matrix

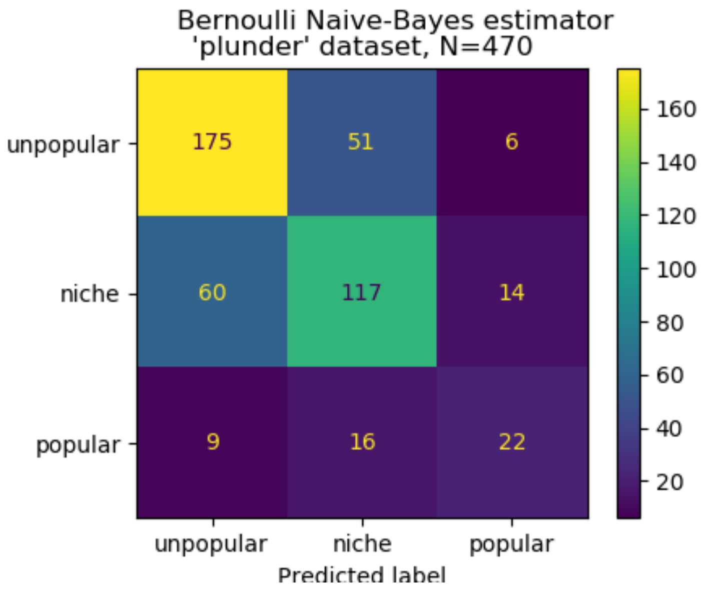


Figure 9: Bernoulli Naive-Bayes Confusion Matrix

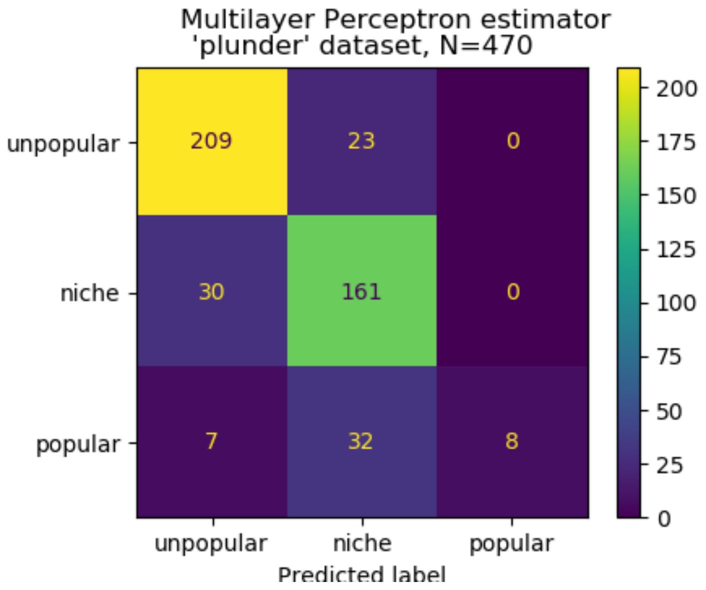


Figure 10: Multi-Layer Perceptron Confusion Matrix

5 Evaluation

5.1 Metric-driven Model Selection

With the previously given metrics in Table 5 and the confusion matrices in Figures 5 to 10, two models arise as promising options based on the full training and the 10-Fold Cross Validation results respectively.

The first model is Support Vector Machines. Boasting an excellent accuracy overall (See Fig. 8) and a remarkable WFPC heuristic metric for full training data, SVC would be expected to accurately predict item classes across game releases. SVC is also expected to accurately recommend the *'popular'* class, and although the intention of the WFPC metric was to credit models recommending *'niche'* items as popular, SVC presents no misclassification of niche items as popular. It is important to note that this logic is based on the fact that the model is trained on the full dataset from a previous release. In particular, for an SVM model with a high value for the C hyperparameter, this will result in overfitting. However, this may be a desirable feature considering this will result in a model that closely represents the environment it was trained in and will probably perform well for game environments similar to the training one. This effect, referred as 'benign overfitting', is a potential factor improving Cross-Release evaluation for One-vs-All Classification SVM variants (Wang, Muthukumar and Thrampoulidis 2021, Bartlett et al. 2020). Additionally, this overfitting process might help represent the implicit bias present in the playerbase's item choices (Shamir 2022).

The second model is Decision Tree. It presents a much higher heuristic metric for the 10-Fold Cross Validation step, but not as big an improvement when fitted to the full dataset as its SVM counterpart. Arnould, Boyer and Scornet (2022) mention overfitting of complex models leading to bad predictive performance to be 'an afforism'. With this in mind, this model is also proposed for debate. Importantly, this is known to be true for fully-grown Decision Trees, where this model is not one of them. Recall Table 1, where the depth of the tree was capped at 20 layers by the Grid Search process. Regardless, this could result in better performance for game releases with vastly different environments, and does not imply bad results for the ones similar to the training environment.

5.2 Cross-Release Evaluation

For Cross Release, both models are expected to generally perform worse. Several factors, both due to data representation as well as external factors impact their accuracy.

5.2.1 Data for new releases

With the goal of assessing if in fact the model can predict newly added items correctly, new datasets are gathered for different, more recent releases of the game. The *'seraph'* and *'lightfall'* datasets, named after the subsequent releases to the base dataset used for all previous training, gathered on the 20th of February and 22nd of March respectively represent these different releases.

A problem present when evaluating the selected model using the newly collected data is the feature space. Not only new items are added, but also perks, which increases the feature space to an incompatible size to the previously trained model. To facilitate evaluation across releases, data is 'retrofitted': The feature space is

Release	**Set**	**WFPC Metric**
Seraph	All	0.07611
	New	0.04040
Lightfall	All	0.07584
	New	0.05194

Table 6: Performance metrics for all Cross-Release Sets. SVC Model

reduced to that of the original training dataset, losing the newly added features in the process. In a way, this is helpful in the case of poor performance since it highlights that the reason an item is popular but is mispredicted by the model is because of this newly added features that were lost due to the retrofitting process.

With all this in mind, the *'seraph'* dataset consists of 519 items and 179 features, while the *'lightfall'* dataset contains 534 items and 185 features.

5.2.2 Performance

An additional influence in the expected performance decrease are external factors affecting popularity. An example of this is balance patches. If item properties and perks are the same values in game, but the game developers decide that globally, a certain weapon type has its stability cut in half, this is not reflected in the data representations, since instead of the values changing, the game interprets them differently. Another might be player strategies, favoring close rather than long-range weapons.

These are also compounding, meaning the later the release from the model's original, performance is expected to be gradually worse.

Confusion matrices for both datasets corresponding to new releases are present in Figures 11 to 18. There is also a distinction present for newly added items, which are expected to perform worse than the entire corpus, since they are likely to rely on newly added perks that are stripped when reducing back to the previous release's feature space.

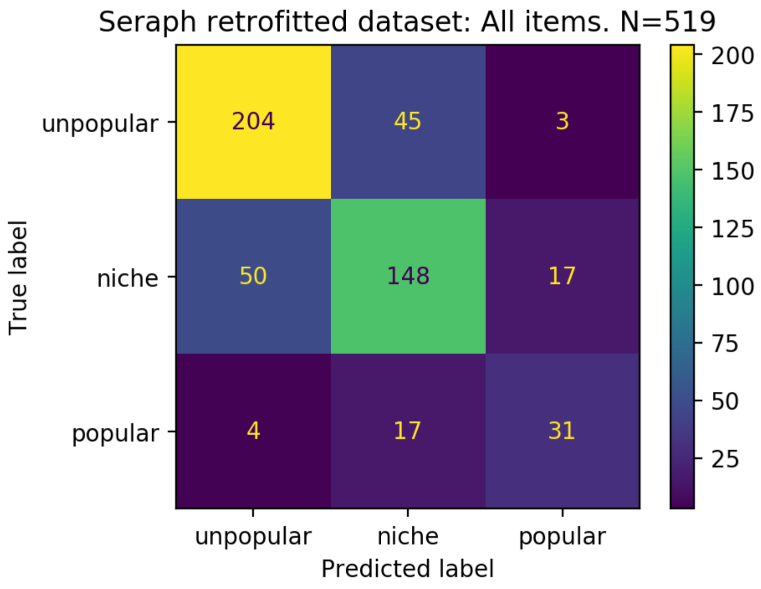


Figure 11: Confusion Matrix for all items in the ’Season of Seraph’ release. SVC Model

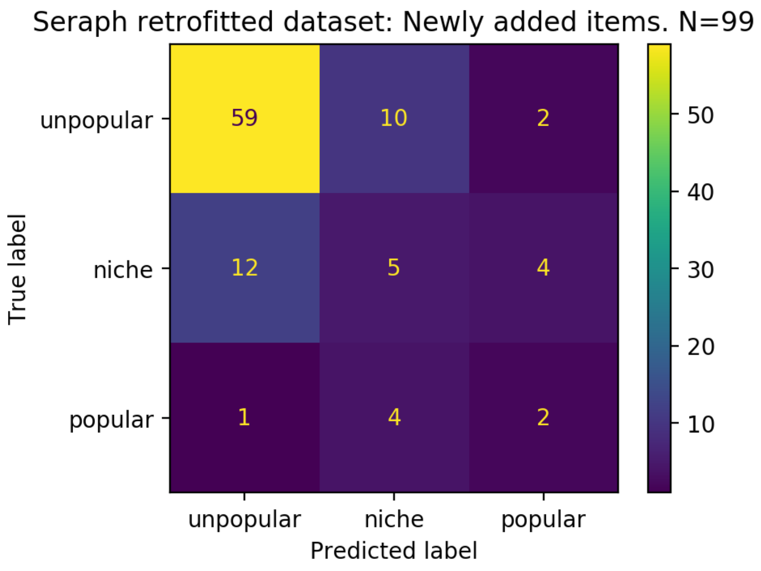


Figure 12: Confusion Matrix for newly-added added items in the ’Season of Seraph’ release. SVC Model

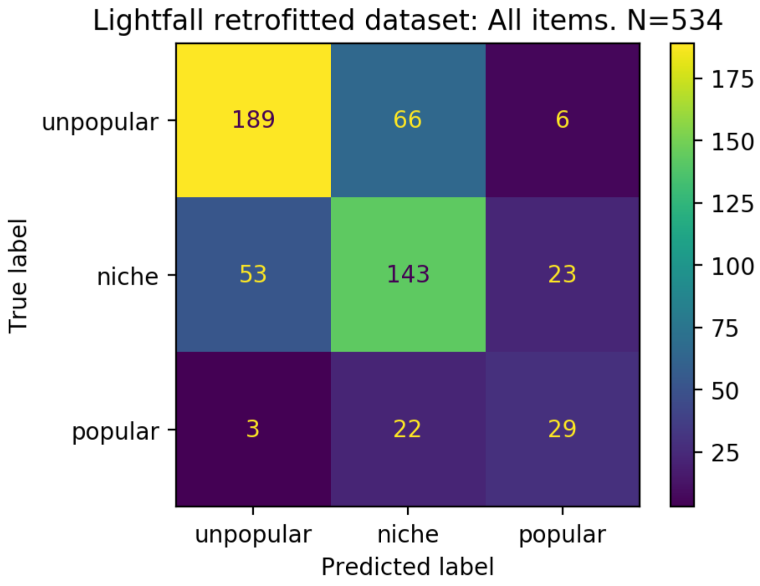


Figure 13: Confusion Matrix for all items in the 'Lightfall' release. SVC Model

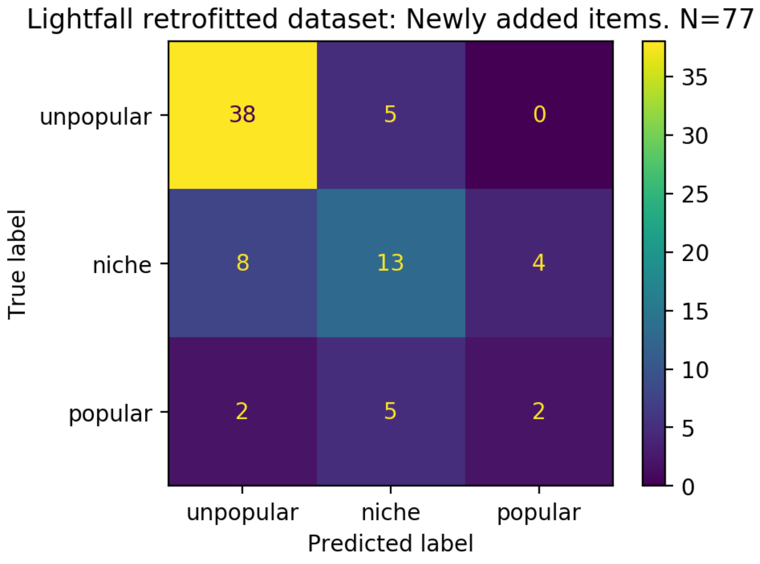


Figure 14: Confusion Matrix for newly-added items in the 'Lightfall' release. SVC Model

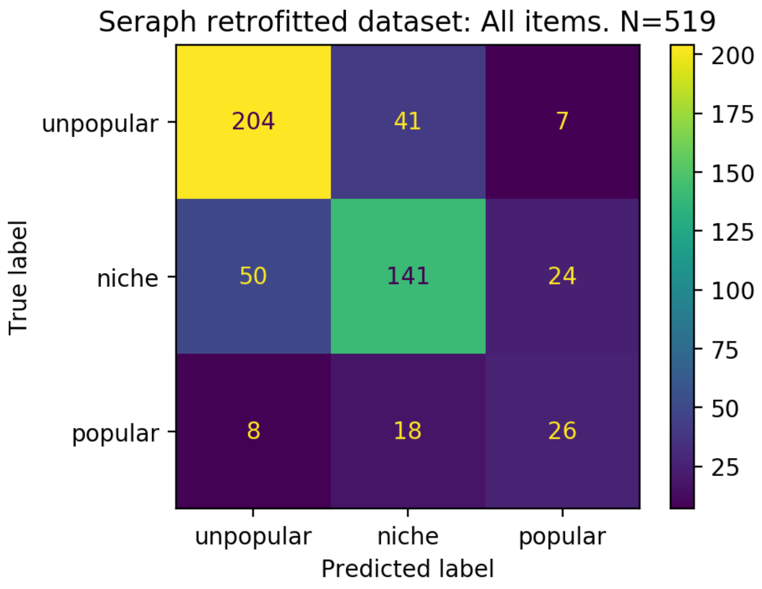


Figure 15: Confusion Matrix for all items in the 'Season of Seraph' release. Decision Tree Model

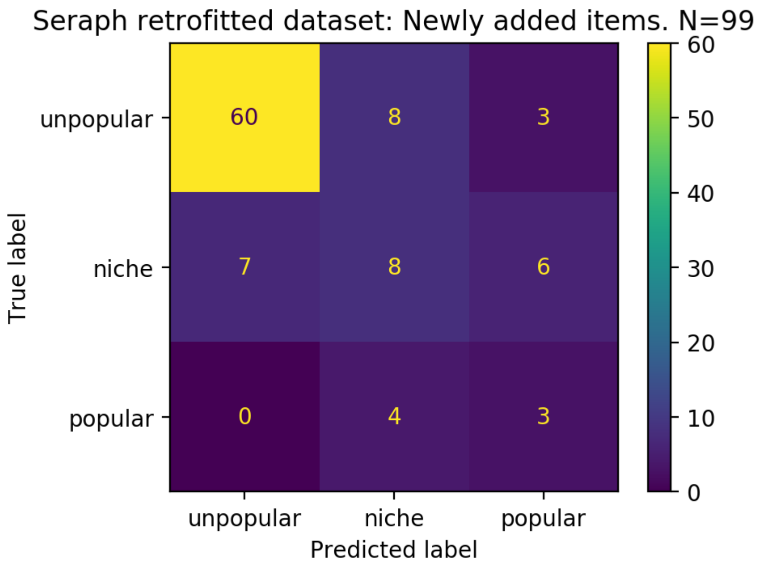


Figure 16: Confusion Matrix for newly-added added items in the 'Season of Seraph' release. Decision Tree Model

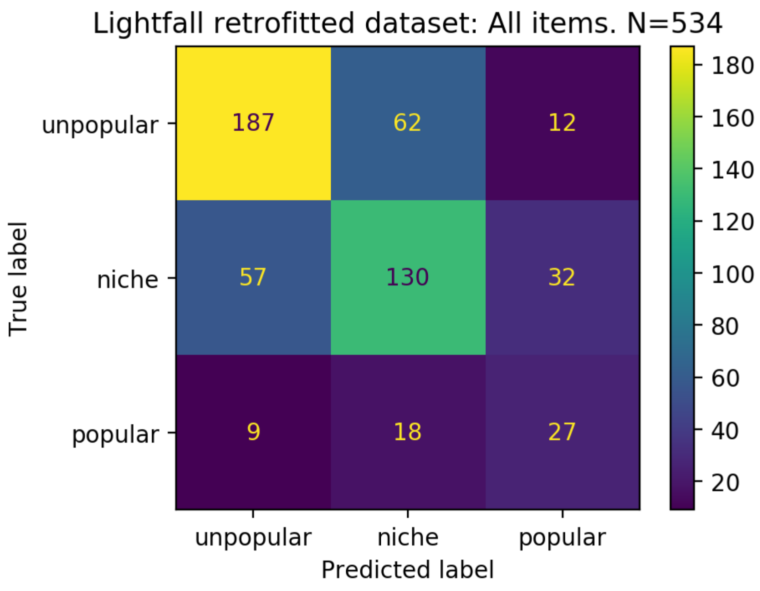


Figure 17: Confusion Matrix for all items in the 'Lightfall' release. Decision Tree Model

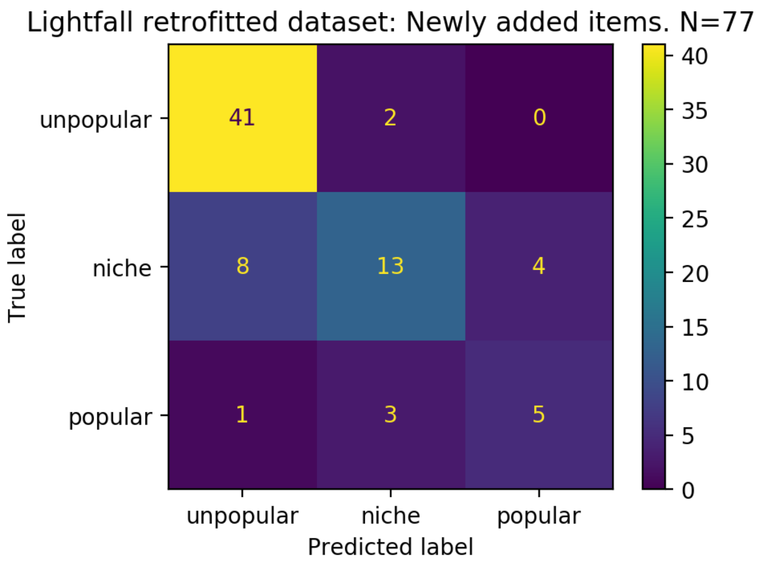


Figure 18: Confusion Matrix for newly-added items in the 'Lightfall' release. Decision Tree Model

Release	**Set**	**WFPC Metric**
Seraph	All	0.07321
	New	0.06060
Lightfall	All	0.08052
	New	0.09090

Table 7: Performance metrics for all Cross-Release Sets. Decision Tree Model

6 Model Choice

As expected, both models perform worse than in their original environment. A point of discussion that may explain this difference is the degree of overfitting and the differences between releases. Recall that the SVM model, when fitted to the full dataset overfits. This is not the case for the Decision Tree model. One would expect the more overfitted model to perform better to game releases closer in time to its original dataset, but this is not the case. In fact, for the *Season of the Seraph* release, which featured an overall small amount of changes as evidenced by the length of its patch notes page[3], lower performance is observed when compared to the *Lightfall* release patch notes[4], a major yearly update revamping several in-game systems and changing several balance aspects relating to items.

Surprisingly, this lower performance for the closer release occurs also for the more generalizable model, the Decision Tree. However, it features better performance in all newly-added sets, and differences in metric no bigger than 0.001 in the complete sets of data. With these results at hand, the Decision Tree Model is chosen for prediction due to the better results in Cross-Release.

A diagram visualization of the Tree is shown in Annex Figure 2a. A result observed in this diagram is that features that players commonly look at for picking their items, such as the Weapon Type and Archetype are not directly used, but inferred from other features. This is due to the multicolinearity present previously mentioned in the data.

[3]`https://www.bungie.net/7/en/News/article/destiny2_update_630`
[4]`https://www.bungie.net/7/en/News/Article/update_7_0_0_1`

7 Bias Report

7.1 Selection Bias

7.1.1 From the data collection process

The match information gathered to obtain the popularity metrics (the category) in the dataset primarily stems from the 'Steam' platform (PC Desktop) due to the use of a 'Steam' platform user as the 'seed' for the snowballing search. Additionally, the 'Aim Assistance' feature is known to have a bigger impact on items when used on a Console platform ('PlayStation' or 'Xbox'), meaning items with higher 'Aim Assistance' values may not be as popular in other platforms. (Mercules904 2023)

Together, this means the class categorization reflects mostly the views of PC Desktop players, and may not give accurate recommendations for players in these other platforms.

Additionally, due to the fact that the representation of an item includes all possible traits for it, some bias could appear on the model, deciding a class as 'Unpopular' based on, for example, the absence of a trait, when in reality the item does not necessarily have to be used with this trait.

7.1.2 From the game's internal match-making

The game finds opponents to play against prioritizing players with better connection using Connection-Based MatchMaking (CBMM). For the snowballing search performed, this means the data was gathered mainly from players in the European region, but unlike with platform there is no way of knowing the location of a player so this is speculation, although likely.

7.2 Automation Bias

7.2.1 From the API reporting

Due to inner workings of the API used to retrieve match information, only items that got at least one defeat or assist on an opponent are reported. This means that an item could be equipped more frequently than shown (i.e. Popular), but it was not used or the player did not manage to get in combat with them so it was not reported. This is especially true for items using certain types of ammunition, which is more scarce.

7.3 Group Attribution Bias

7.3.1 From items with certain features

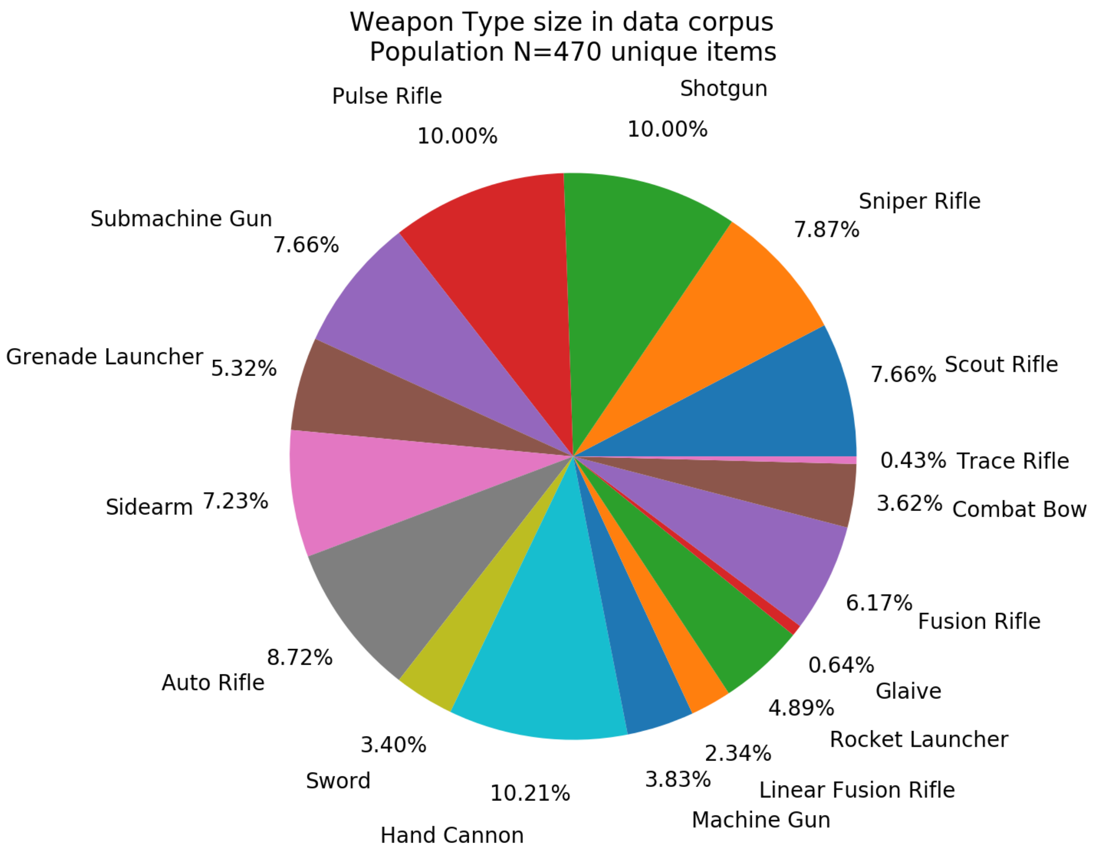


Figure 19: Share of the dataset for all Weapon Types. *'Plunder'* dataset

In Fig. 19 the share of each Weapon Type on the dataset is represented. It can be seen that some item types are underrepresented, like ‘Glaive’ and ‘Trace Rifle’. This is due to the fact these item types have been introduced in the game relatively recently and therefore not as many items of that type exist yet in game.

This might mean popular items belonging to those item categories may be misclassified simply because there was not as much sample data for the model to learn.

Additionally, players appear or revive in a match with infinite Primary weapon ammunition, limited Special ammo, needing to obtain it off of other player's drops when they defeat them, and no Power/Heavy ammo, of which pickups appear at set intervals during the game and are available for everyone during 30s after pickup.

In short, Primary ammo weapons are more readily available than Special ammo weapons, and Heavy weapons are very limited in usage, meaning the majority of usage comes from Primary weapons. Fig. 20 is a chart of representation of items grouped into the three ammunition types:

As expected, the majority of items recorded were Primary ammo weapons. The implications for predictions is that Special and Power ammo items are underrepresented when compared to Primary, and prediction could be less accurate for these less frequent weapons, which as discussed previously are in turn split into their own item types. For instance, for underrepresented item types that also use one of the less available types of ammunition, this means even less samples of these types are obtained, and prediction for them could be even less accurate. Below is an example to illustrate how within one of the less available types of ammunition, items are already at a disadvantage.

This results in item types such as the previously mentioned 'Glaive' and 'Trace Rifle' types, their use of the less available 'Special' ammunition, compounds down the likelihood of them appearing in games.

7.4 Impact of and Conclusion to the Bias Assessment

Given the problem context, being video-game item recommendations, the impact of misprediction is negligible. As discussed on the 'Performance Evaluation' section, Type I errors could even be considered as beneficial to certain models if they are more likely to classify a 'Niche' item as 'Popular', resulting in the introduction of the WFPC heuristic metric.

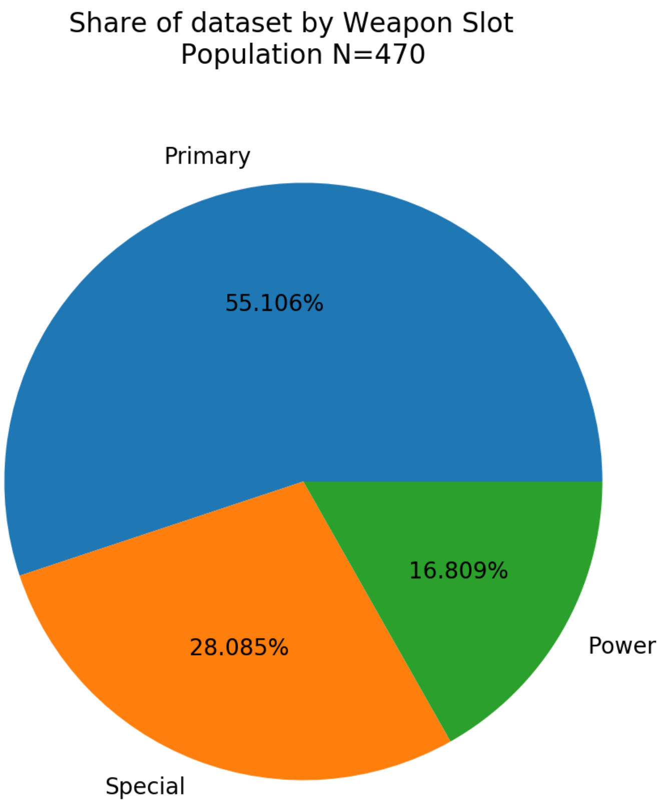


Figure 20: Share of the dataset of Weapon Types grouped by ammunition type. *'Plunder'* dataset

In real cases, the worst scenario that may happen is that a player is recommended an item as 'Popular' when in reality it is not, and they do not realize and keep using it. Although item popularity typically corresponds to high performance, the main driving factor of success in games is a player's skill, so this may not even impact their in-game performance, or have only slightly negative impact.

Another scenario to discuss is when the model misclassifies a 'Popular' item as any other category. Since the intended use is to only recommend popular items, it would not recommend the item at all. Falling out of the intended use, Type II errors are considered negligible.

Summarizing, in terms of biases, the data used to train the model presents biases towards the items used in PC Desktop platforms and in Europe. It may underperform for items types with low representation in the overall pool of items. Finally, the category of certain data in training might have been inaccurate.

The impact of misprediction is assessed to be very low due to the problem domain and intended use of the model.

8 Model Card

Following the template laid out by Mitchell et al. (2018)

Model Details

- **Person or Organization developing the model**
  Jorge Jiménez García, for Technological University Dublin, v0.1

- **Model Date** 30 April 2023

- **Model Type** Decision Tree

Intended Use

- Intended to recommend newly released items in the video-game *'Destiny 2'* for future game versions (i.e: Cross-Release)
- Not intended to classify all popularity types

Factors

- Potential relevant factors include an item's Weapon Type, the presence of new perks or attributes not part of the previous release's feature space

Metrics

- **Evaluation Metrics** consist of a heuristic measure of the accuracy of prediction for *'Popular'* items (See Equation 1). Balanced Accuracy was also taken into consideration. Experimental performance for the intended use case was also considered
- **Confusion Matrices** See Figure 5 for Training and Figures 15 to 18 for Cross-Release.

Training *'Season of Plunder'* dataset. 10-Fold Cross Validation training splits.

Evaluation

- *'Season of Plunder* dataset. 10-Fold Cross Validation evaluation splits.
- *'Season of Seraph'* dataset. Both complete and filtered for newly added items.
- *'Lightfall'* dataset. Both complete and filtered for newly added items.

Ethical Considerations

- Due to the search strategy, classes assigned to items may reflect the views of mostly European players from the PC platform.

Caveats and Recommendations

- The Cross-Release prediction may underperform due to external changes to the game that the model cannot account for
- The Cross-Release prediction may underperform when an item presents properties not present on the original training data's feature space.

Addendum A visualization of the model can be seen in Annex Figure 2a

9 Production

To surface this information and help users better choose the items they use, an operational system is envisioned that would continuously gather data to form more accurate popularity metrics. Given that new items generally come with new game releases, which typically occur every three months, the system is free to gather data for a long period of time. It would then be fitted for the current environment close to the new release's date, prepared to acquire the new list of items on release and rank newly released items into their corresponding categories. Additionally, since the model is based in past releases, all previous models could be stored, allowing comparisons between predictors.

One of the ways this could be made accessible is through a website dashboard that would display the list of newly added items sorted into their popularity class. This being a recommendation system, popular items, i.e, the class of most importance, would be displayed first for users, and would allow them to filter for other specific features they might be looking for in the newly released items. Using the aforementioned models from previous releases, users could also select a previous game release, tied to a model, that they may think closely resembles the current, unseen one to get more tailored recommendations. Via this system, players could also be able to select a past environment they particularly enjoyed to suit their recommendation towards their particular preferences.

10 Conclusion and Future Work

10.1 Conclusion

To finalize, a summary on the work done. A data gathering algorithm was proposed and executed to collect information on in-game items and their frequency of appearance in different matches. From there, a representation was built from data present in the game's public database and fed into six different Machine Learning models for evaluation. A new approach to evaluation was developed to take advantage of the particularities of recommendation systems: only recommending the top or popular items.

These models were tuned using a Grid Search process and evaluated using this heuristic metric using 10-Fold Stratified Cross Validation and then again on the entire corpus of data. Once reduced the options to only two models, they were evaluated again in their 'live' or intended use case: prediction of newly released items into the game. Challenges such as the change in feature space across releases were overcome, and in live evaluation a Decision Tree model was chosen as the best performing. Once selected, potential biases and their sources were addressed, and from these results a Model Card was redacted.

Overall, very satisfactory results were obtained from live evaluation, as well as an unexpected improvement in performance for releases that differ significantly from the model's.

The use of a heuristic metric is also proven to work reliably in the ambit of popularity prediction for recommender systems.

10.2 Future Work

For proposed improvement to the recommender systems, ways of incorporating the outside factors modified across releases into the model is of interest. One such approach could be the incorporation of an adaption layer, that would take in the prediction of a previous model and incorporate the features that are not a part of the original model back into the prediction. The difficulty would lie in weighing or training the adaption layer with these newly-added, unseen features to score them correctly. A technique that could be of use is Natural Language Processing, reading in descriptions of the newly added feature and outputting an estimation of how important it may be for the popularity of an item.

An additional proposed line of work would be the acquisition of more detailed data. Due to the API design, only an online user can see the exact details of the items they are equipping at the present time. No information about items equipped in the past is kept. Having authenticated access from several users would allow a drastic increase in the amount of data available, which would lead to not only an improvement on the accuracy of predictions, but also on the specificity of predictions, where the model would now be able to say if a *particular instance* or roll of an item is good, instead of only being able to classify the item in general. A challenge with this is the collection of accurate target variables. With such a wide range of items and instances, it is hard to come by an exact copy of an instance more than once so that its popularity can be gauged. A different approach to determining the target variable would be needed.

Another suggested feature upgrade would be the development of a 'loadout auto-complete'. In essence, by reading in all gathered loadout combinations, which recall are formed by three items. Given an item a user wants to equip, the rest of the loadout could be suggested to them. A challenge with this proposal is data gathering, since although three items are equipped at all times in a loadout, the design of the API only reports items that are used in a match, meaning some items may go unreported. However, this may show improvement on predictions since with every loadout being a unique instance, there is no limit on the data gathered unlike with the previous sets where the limit was the number of unique items present in-game.

Additionally, work on the heuristic metric used for evaluation could be of benefit. In a similar way to recall and precision having to be used in tandem for evaluation, resulting in the F_1-score metric, the heuristic could be in a way combined with a different metric such as accuracy, with the goal of, for the heuristic side, crediting failed but only slightly off predictions while on the accuracy side, ensuring not too many of the completely undesirable class slip past. This would also help in obtaining a unified metric for evaluation.

A different way of improving the heuristic metrics is the study of appropriate factors to the terms of the metric to more accurately reflect the intended behaviour. For instance, the term on FP_{Niche} could be adjusted to have diminishing returns, where a model with, for instance 20 Niche False Positives and 0 True Positives would not equate to a model with 10 True Positives and no False Positives.

Finally, a proposed improvement to the model would be accurately pruning the tree to lesser depth. As shown by Annex Figure 2a, the tree is left-heavy with the majority of the deep nodes concentrating on that side of the tree. Importantly, the mentioned branches only cover 'Unpopular' and 'Niche' items, so a way of reducing the complexity of the tree for these less relevant classes would be of benefit. A challenge for this is ensuring the terminal nodes on that branch of the tree do not worsen performance for the overall model.

References

Arnould, Ludovic, Claire Boyer and Erwan Scornet (2022). 'Is interpolation benign for random forests?' In: *arXiv preprint arXiv:2202.03688*.

Bartlett, Peter L et al. (2020). 'Benign overfitting in linear regression'. In: *Proceedings of the National Academy of Sciences* 117.48, pp. 30063–30070.

Bertens, Paul et al. (2018). 'A Machine-Learning Item Recommendation System for Video Games'. In: *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 1–4. DOI: 10.1109/CIG.2018.8490456.

Blanco-Justicia, Alberto and Josep Domingo-Ferrer (2019). 'Machine learning explainability through comprehensible decision trees'. In: *Machine Learning and Knowledge Extraction: Third IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2019, Canterbury, UK, August 26–29, 2019, Proceedings 3*. Springer, pp. 15–26.

Drachen, Anders et al. (2016). 'Guns and guardians: Comparative cluster analysis and behavioral profiling in destiny'. In: *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 1–8. DOI: 10.1109/CIG.2016.7860423.

Fowler, Erin E. et al. (2020). 'Empirically-derived synthetic populations to mitigate small sample sizes'. In: *Journal of Biomedical Informatics* 105, p. 103408. ISSN: 1532-0464. DOI: https://doi.org/10.1016/j.jbi.2020.103408. URL: https://www.sciencedirect.com/science/article/pii/S1532046420300368.

Jaderberg, Max et al. (2017). 'Population based training of neural networks'. In: *arXiv preprint arXiv:1711.09846*.

Joshi, Rishabh et al. (2019). 'A Team Based Player Versus Player Recommender Systems Framework For Player Improvement'. In: *Proceedings of the Australasian Computer Science Week Multiconference*. ACSW '19. Sydney, NSW, Australia: Association for Computing Machinery. ISBN: 9781450366038. DOI: 10.1145/3290688.3290750. URL: https://doi.org/10.1145/3290688.3290750.

Kosztyán, Zsolt T, Marcell T Kurbucz and Attila I Katona (2022). 'Network-based dimensionality reduction of high-dimensional, low-sample-size datasets'. In: *Knowledge-Based Systems* 251, p. 109180.

Liu, Bo et al. (2017). 'Deep Neural Networks for High Dimension, Low Sample Size Data.' In: *IJCAI*, pp. 2287–2293.

Looi, Wenli et al. (2019). 'Recommender System for Items in Dota 2'. In: *IEEE Transactions on Games* 11.4, pp. 396–404. DOI: `10.1109/TG.2018.2844121`.

Mercules904 (2023). *How do Aim Assist and Accuracy Work in Destiny 2?* URL: `https://www.destinymassivebreakdowns.com/blog/aimassist` (visited on 18/04/2023).

Mitchell, Margaret et al. (2018). 'Model Cards for Model Reporting'. In: *CoRR* abs/1810.03993. arXiv: `1810.03993`. URL: `http://arxiv.org/abs/1810.03993`.

Mukherjee, Amitangshu, Isha Garg and Kaushik Roy (2023). 'Encoding hierarchical information in neural networks helps in subpopulation shift'. In: *IEEE Transactions on Artificial Intelligence*.

Pedregosa, F. et al. (2011). 'Scikit-learn: Machine Learning in Python'. In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Probst, Philipp and Anne-Laure Boulesteix (2017). 'To tune or not to tune the number of trees in random forest'. In: *The Journal of Machine Learning Research* 18.1, pp. 6673–6690.

Sepiolo, Dominik and Antoni Ligęza (2022). 'Towards Explainability of Tree-Based Ensemble Models. A Critical Overview'. In: *New Advances in Dependability of Networks and Systems: Proceedings of the Seventeenth International Conference on Dependability of Computer Systems DepCoS-RELCOMEX, June 27–July 1, 2022, Wrocław, Poland*. Springer, pp. 287–296.

Shamir, Ohad (2022). 'The implicit bias of benign overfitting'. In: *Conference on Learning Theory*. PMLR, pp. 448–478.

Trzciński, Tomasz and Przemysław Rokita (2017). 'Predicting popularity of online videos using support vector regression'. In: *IEEE Transactions on Multimedia* 19.11, pp. 2561–2570.

Wang, Ke, Vidya Muthukumar and Christos Thrampoulidis (2021). 'Benign overfitting in multiclass classification: All roads lead to interpolation'. In: *Advances in Neural Information Processing Systems* 34, pp. 24164–24179.

Wu, Shaomin, Peter Flach and Cesar Ferri (2007). 'An improved model selection heuristic for AUC'. In: *Machine Learning: ECML 2007: 18th European Confer-*

ence on Machine Learning, Warsaw, Poland, September 17-21, 2007. Proceedings 18. Springer, pp. 478–489.

Annex

Figures

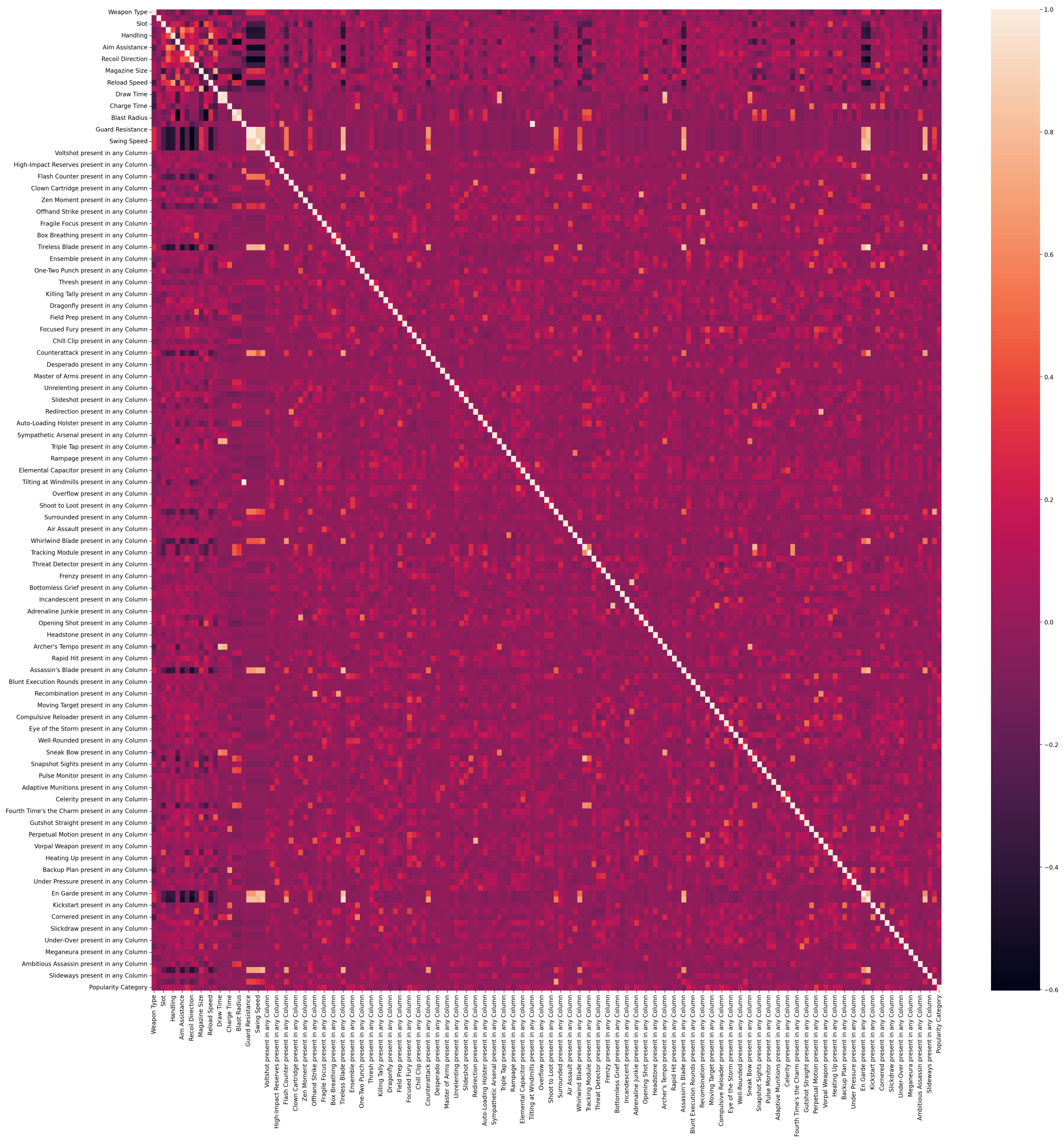


Figure 1: Correlation heatmap for all features in the dataset

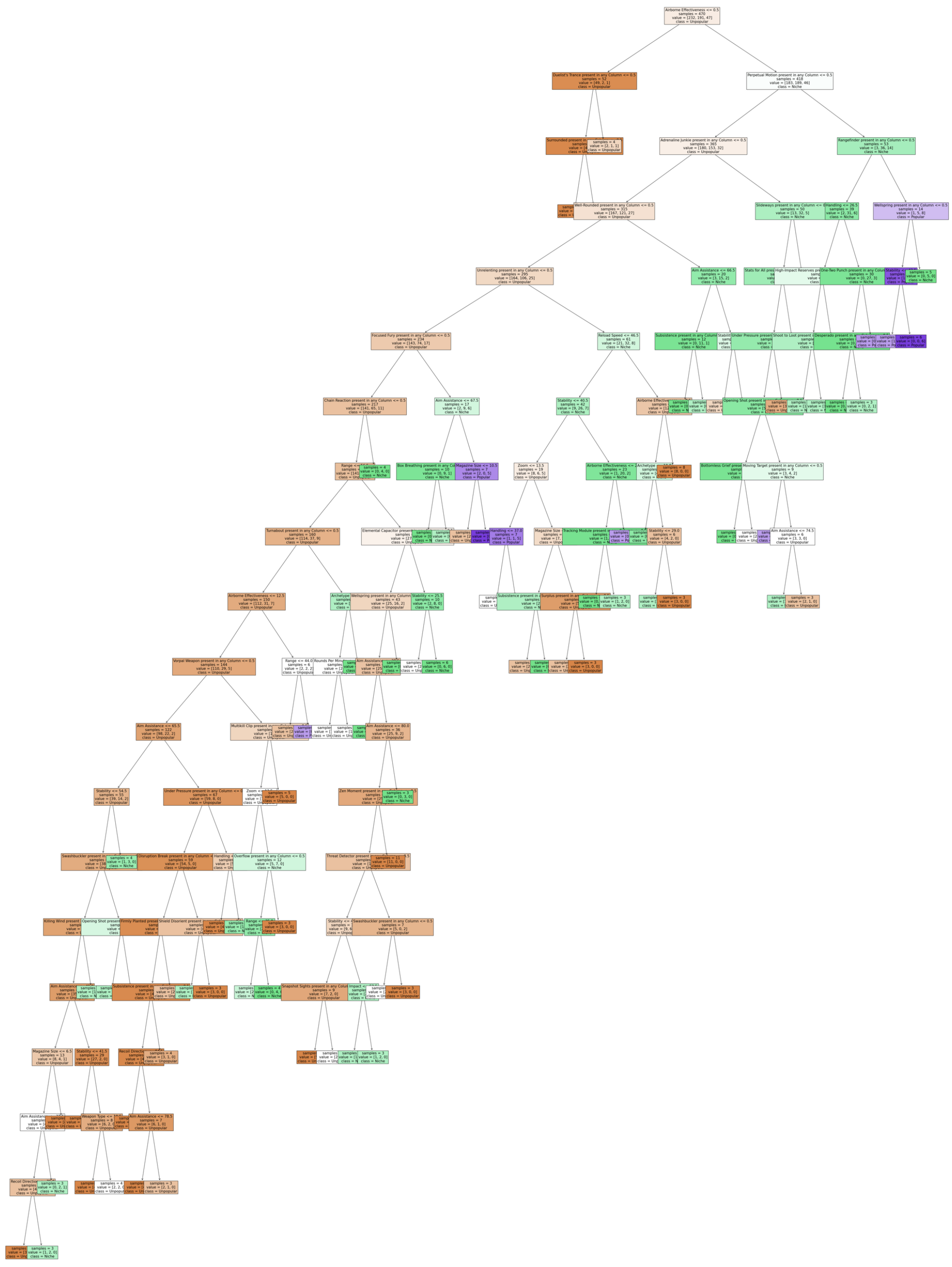


Figure 2: Final Decision Tree Layout.

(a) Orange, Green and Purple nodes represent 'Unpopular', 'Niche' and 'Popular' classes respectively

Tables

Name of Feature	**Description**	**Type, [range/unique values]**	**Notes**
Weapon Type	The type of weapon this item is	Categorical [17]	
Archetype	Modifies how it fires and describes its stat profile	Categorical [15]	Indirectly describes Stability, Handling, Range and Fire Rate
Slot	The type of ammunition the weapon uses	Categorical [3]	
Stability	The bounce intensity and direction your weapon experiences when fired.	Numerical [0,100]	except Swords and Glaives
Handling	The speed with which the weapon can be readied and aimed.	Numerical [0,100]	except Sword and Bow
Range	Increases the effective range of this weapon.	Numerical [0,100]	except Swords, Bows, Rocket and Grenade Launchers
Aim Assistance	Determines the bullet magnetism of your shots.	Numerical [0,100]	except Swords
Airborne Effectiveness	How effective this weapon is in the air.	Numerical [0,100]	except Swords
Recoil Direction	An expression of a Weapon's recoil path	Numerical [0,100]	except Sword.
Zoom	Magnification and effective range increase when aiming down sights.	Numerical [0,100]	except Swords and Glaives
Magazine Size	How many rounds you can fire before reloading.	Numerical [1, inf]	except Bows
Impact	Increases the damage inflicted by each round.	Numerical [1,100]	except Rocket and Grenade Launchers
Reload Speed	The time it takes to reload this weapon.	Numerical [0,100]	except Swords
Rounds Per Minute	How fast this weapon fires.	Numerical [15,1000]	except Swords
Draw Time	Determines how quickly you can charge an arrow.	Numerical [0,100]	only on Bows
Accuracy	How well shots fired by this weapon hit their target.	Numerical [0,100]	only on Bows
Charge Time	Determines how quickly this weapon can fire.	Numerical [20,1000]	only applicable on Fusion Rifles
Velocity	Increases the speed of projectiles fired by this weapon.	Numerical [0,100]	only on Rocket and Grenade Launchers
Blast Radius	Increases the explosion radius of this weapon.	Numerical [0,100]	limited to 55 for Grenade Launchers
Shield Duration	How long you can maintain your guard with this weapon.	Numerical [0,100]	only present on Glaives
Guard Endurance	How long you can maintain your guard with this weapon.	Numerical [0,100]	only present on Swords
Guard Efficiency	Reduces the amount of energy required to guard an attack.	Numerical [0,100]	only on Swords
Guard Resistance	Damage reduction while guarding with this weapon against most attacks.	Numerical [0,100]	only on Swords
Swing Speed	How fast you can swing this weapon.	Numerical [0,100]	only on Swords
Charge Rate	How fast this weapon's energy recharges.	Numerical [0,100]	only on Swords
$\boldsymbol{X}$ present	If perk $\boldsymbol{X}$ is present on any of the weapon's two columns	Boolean [149]	Some perks are only on certain Weapon Types

Table 1: Description of common features and their effect on an item. Taken from in-game descriptions