
Fighting Game Balance Through Machine Learning & Feature Engineering

Nick Chen
nc5@princeton.edu

Abstract

This paper presents a baseline study for analyzing the structure and balance of fighting video games. Feature engineering and machine learning methods are applied to a data set describing characters from the Super Smash Bros. Ultimate video game, where the characters' properties and frame data are the features, and their ratings on a scale of 0.0 to 5.0, estimated by a group of professional players, is the target. Because the data set is very small, regression methods produced heavily overfitted results, not only due to a minimal number of samples, but also because the scale of target ratings was entirely subjective. Despite this, the initial models were partially successful at determining which characters were better than others, i.e. order of "goodness" between characters was largely preserved. To address these issues, the task of predicting a character's overall performance was redefined as a classification problem, where the target was a character's "tier," A through E, as opposed to an exact numeric value. This change resulted in significantly reduced variance in the model. This analysis as a whole demonstrates the potential for creating an accurate model despite small data, through feature engineering.

1 Introduction

Game balance is one of the most difficult tasks facing video game developers today. As competitive video games and "Esports" become increasingly popular and lucrative, developers must ensure that gameplay is fair and clearly structured, such that their games remain viable arenas for professional competition. In addition, it has been theorized that much of a game's popularity, marketability, and longevity rely on its appeal to the competitive scene, i.e. a well balanced game with a large and enthusiastic competitive fanbase will sell more copies [7]. Therefore, effective game balance is key to creating enjoyable, marketable, and long-lasting video games.

David Sirlin, a professional gamer and developer, insists that game balance is best achieved through extensive manual testing, and that changes must be made upon human intuition: "Game balance is so complex as to be inherently unsolvable Intuition, not math, is the best tool to navigate high-complexity problems" [7]. Sirlin swears by traditional methods of game testing, whereby developers and testers play the game over long periods of time, so that they develop a sense of what aspects of the game are fair/unfair, and can make appropriate changes [7]. Sirlin rejects computational/quantitative methods of game balance, not because he refutes math and science, but because a player's enjoyment is derived from a wide variety of gameplay options that can only be created through added complexity, such that the game is "unsolvable" [7].

However, Sirlin also admits that game balance, through the process of intuitive data collection he describes, requires feedback from developers and professional players over long periods of time [7]. Because of this, rifts between a game's developers and its competitive fanbase can form very easily, as complaints from players that certain aspects of the game are unfair motivates quick developer edits and updates, informed only by short-term intuition, as opposed to concrete methods of analysis.

Of course, devising a more consistent method of balancing itself is challenging, so it seems more practical to stick to human testing.

Despite Sirlin’s assertions against mathematical analysis, the study presented in this paper attempts to solve the issues surrounding subjective human play testing through statistical and machine learning methods, in order to inform better game balance decisions. This study focuses on the game Super Smash Bros. Ultimate (Smash Ultimate), currently one of the most popular competitive fighting games. Having gone through multiple versions and development updates over its 20 year lifespan, the Super Smash Bros. franchise provides an opportunity for combining the longterm intuitive expertise of professional players with quantitative analysis. Furthermore, the game boasts a roster of over 70 characters, which is more than any other relevant fighting game. From a machine learning perspective, this is an unfortunately small data set, which makes the task even more difficult.

To address the issue of few samples, I performed extensive feature engineering and data organization (see 2.2), so that the machine learning methods would yield the best possible results. My analysis was designed such that each sample was an individual character, whose moves, attributes, properties, etc. were recorded and summarized as features, and their rating (i.e. how “good” or “bad” they are in the game) was the target (a detailed description of this rating value is described in 2.1). The purpose of analyzing the data in this way was to create a model for predicting a character’s performance given their base characteristics (independent of the player and their play style, dexterity, mood, etc.), such that developers could balance characters that perform especially well or poorly by editing specific features. Additionally, a successful model would allow developers to present the features of hypothetical characters and immediately determine how well they will perform.

To build a model, a variety of regression methods, particularly the Random Forest algorithm, were trained on a randomly selected majority portion of the data (50 samples or more), and used to predict rating values for the rest (see 4.1). Results were mediocre in most cases, with a maximum coefficient of determination of around 0.27. Eventually, the model was changed to a classification problem, which yielded improved results (mean accuracy of around 0.41; see 4.2).

2 The Data

One of the biggest challenges of analyzing game balance for a player such as myself is finding and collecting the relevant data required for meaningful research. Naturally, developers are often reluctant to release the data for their games, as this diminishes the discovery potential of the game’s intricacies, and therefore hurts its lifespan (see [9] for an example). Therefore, players take it upon themselves to record this data manually. Data collection is especially important for fighting gamers, as winning in fighting games is heavily reliant on the precise timing of moves and combos, so players in the competitive scene will painstakingly record the timing and properties of every move of every character (also known as “frame data,” since time is measured in display frames). Thus, most of the data used in this study is community-sourced, and pulled from a conglomerate of public internet references. The validity of such data is justified by virtue of the Smash Ultimate community’s large size, consistent updating, correcting, and cross-referencing of the data, and general enthusiasm for exciting and fair gameplay.

2.1 Data Sources

The three main sources of data I used came from the Kurogane Hammer website, and users from Reddit and the online forum Smashboards [2, 4, 5]. Kurogane Hammer is run by a single user, and provides data for character attributes (the website also presents frame data, but it is incomplete) [5]. Character attributes are properties such as walk/run speed, air velocity, weight, essentially the most visually apparent characteristics. Additional attribute data for maximum jump height, as well as recovery distance, were sourced from the SmashWiki website and a Facebook page called “Untitled Animation,” respectively [8, 10] (I also recorded some of this data myself, as it was missing from these sources; see Appendix A). Frame data was pulled from a Smashboards user who goes by the alias “Zapp Branniglenn” [2]. Branniglen has recorded frame data for every character in Smash Ultimate, which is frequently updated and cross-referenced by other users [2]. Lastly, target data was pulled from a Reddit user who goes by the alias “inktivate” [4]. This user constructed a comprehensive tier list for Smash Ultimate characters (i.e. how well characters perform relative

to each other) by combining individual tier lists created by several professional and well-respected players [4]. Although these tier lists are entirely based on opinion, it can be argued that these pro players have, through extensive gameplay, developed an "intuitive" sense of what makes a character good or bad. While this does not mean their intuition can be interpreted as law, it provides us a solid reference for measuring our target variable (for more on how the issue of subjectivity was addressed, see 4.2).

2.2 Features & Feature Construction

Features for each character included their base attributes and frame data. Processing attribute data was very simple, as each attribute was recorded as a single feature. Furthermore, very general correlations could be made between some of these basic attributes and a character's rank, as shown in Figure 1 below. While these attributes are clearly not the only indicators of a character's rating, this simple initial data analysis shows that there is latent structure in character features, i.e. it is possible to develop reasonable rating predictions from feature data.

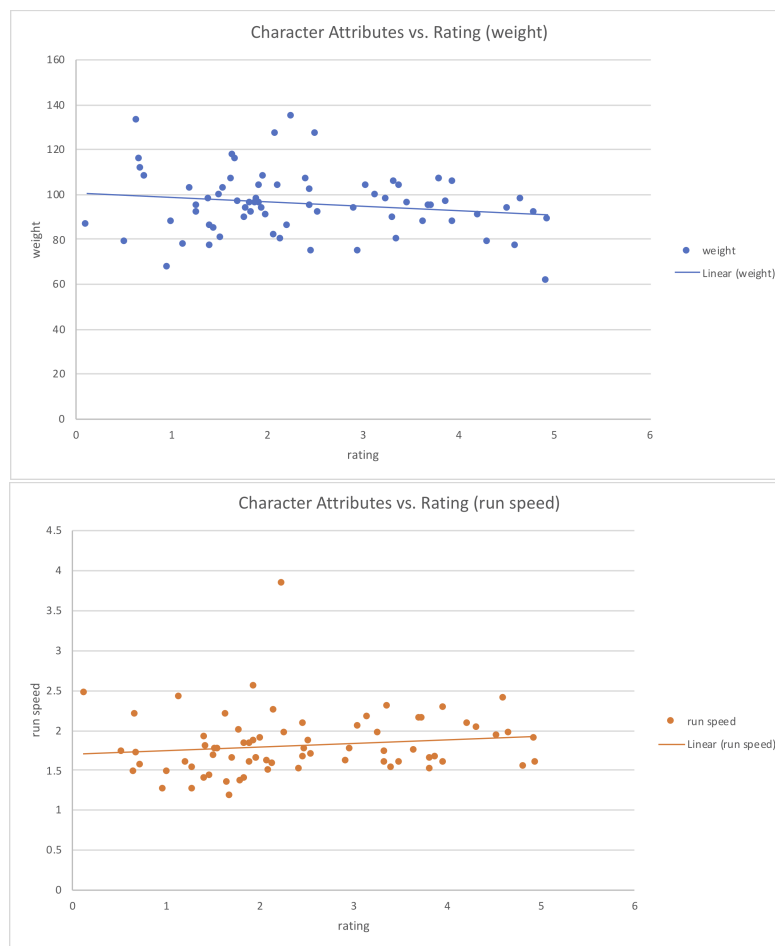


Figure 1: Character Attribute Examples vs. Rating

On the other hand, frame data was more difficult to collect and organize as features. Every character has a multitude of different moves and combinations of moves, some of which are unknown, or can only be performed against specific characters. Furthermore, the moves themselves have various properties. Therefore, I decided to input only the most important features from the available data: a specific set of moves common between all characters, their base damage, startup (the number of frames it takes for the move to deliver the initial hit), and advantage (the number of frames that the character has to execute another move before the opponent can retaliate). Summarizing the frame data in this way would hopefully capture the speed, power, and combo potential of every character's

moves. Despite the commonality of these features between characters, cleaning this data took a significant amount of time, as some moves change based on specific button inputs or myriad other situational properties (for more details on game-specific modifications to the data, see Appendix A).

Another important aspect of Smash characters besides their basic attributes and move speed/power is their ability to "recover," or return to the fighting platform after being knocked into the air by an opponent. Although every character has a designated recovery move that usually sends them flying upward, recovery is still a very difficult concept to quantify, as it is characterized more by the number of options a character might have in a given situation than the distance they can travel to return to the platform. I decided on a few parameters to measure recovery: top hitbox, invulnerability, maximum vertical distance, and horizontal movement. Top hitbox refers to whether or not a character has a hitbox at the top of their vertical recovery move (it is harder to hit an ascending character if they can hit you at the same time); invulnerability refers to whether or not a character is invulnerable at some point while performing their recovery move; maximum vertical distance is the farthest in-game vertical distance a character can travel (this was optimized and recorded in a video posted on the "Untitled Animation" Facebook page [10]; multiple other sources were consulted, but I decided this was the best one to use); horizontal movement is a multi-class variable measuring how well a character can cover horizontal distance using their recovery move: 1 = can move almost completely horizontally, 0 = can partially angle the move, -1 = only moves vertically (given that I've played the game for years and also own a copy, I was able to record this data myself). Some characters also use ropes or tethers to latch onto the platform from a distance, so I recorded whether or not each character has a tether move available to them. I also added the maximum number of jumps each character could perform (I recorded this data myself).

In addition, a few extra variables were added to account for other key characteristics. One of the major limitations of Branniglenn's frame data is a lack of hit and hurt box reference. The hit boxes of moves, or the area of effect that moves cover when executed, as well as characters' hurt boxes, or the area where an opponent's move can make contact with the character, are extremely important for a character's performance. For example, characters who use weapons or tools that extend outside their bodies are generally better, as their weapons have hit boxes but no hurt box. A proxy variable for hit/hurt box was created, which tells whether or not a character primarily uses an external weapon. Lastly, I created proxy variables for some "special moves" of the characters. Specials are moves with complex properties and mechanics that are very specific to each individual character, which makes them very difficult to compare between samples. However, many characters have similar types of specials, such as projectiles, or counters/reflectors that turn opponents' moves against them. Knowing this, I recorded whether or not each character has an available counter, reflector, and/or projectile (projectiles are also present in some of the moves for which I included more extensive data; this variable includes those). Below is an example of the full data for one character (see Figure 2). Despite the extensive organization, construction, and cleaning described above, the current data is still limited in describing the full capabilities and flaws of each character (see Section 4 for further discussion of these limitations and their effects on the results). Therefore, it was essential for me to build the data set as thoroughly as possible, given the sparse sample size.


Full Feature Data for Mario (rating = 2.96)											
attributes	air accel	0.08	Jab	2	Jab	-29	Jab	8.9	recovery	top hitbox?	
	air speed	1.21	F-Tilt	5	F-Tilt	-13	F-Tilt	7		invulnerability	
	max fall spee	1.50	U-Tilt	5	U-Tilt	-18	U-Tilt	5.5		max vertical	12
	fast fall spee	2.40	D-Tilt	5	D-Tilt	-15	D-Tilt	7		horizontal m	0
	initial dash	1.94	Dash Attack	6	Dash Attack	-17	Dash Attack	8	other	tether?	0
	run speed	1.76	F-Smash	15	F-Smash	-22	F-Smash	17.7		projectile?	1
	walk speed	1.16	U-Smash	9	U-Smash	-20	U-Smash	14		reflector?	1
	weight	75.00	D-Smash	5	D-Smash	-31	D-Smash	10		counter?	0
	full hop	36.33	N-Air	3	N-Air	-2	N-Air	8		max jumps	2
	short hop	17.54	F-Air	16	F-Air	-12	F-Air	14		weapon?	0
	air jump	36.33	B-Air	6	B-Air	-2	B-Air	10.5			
			U-Air	4	U-Air	-3	U-Air	7			
			D-Air	5	D-Air	-11	D-Air	12			
			Grab	6			Pummel	1.3			
			Dash Grab	9			Forward Thr	8			
			Pivot Grab	10			Back Throw	11			
							Up Throw	7			
							Down Throw	5			

Figure 2: Example Full Feature Data for One Sample

Training and test sets were separated from the data randomly, with the training sets containing at least 50 samples each. Two separate data sets were tested, one with only attributes and frame data (set A), and another including the extra variables described above (set B). Over 20 different training and test sets were generated and used to build models using the regression methods described below.

3 Machine Learning Methods

Choosing the right estimator for such a small data set was tricky. I decided to use the Random Forest algorithm, implemented through sci-kit-learn.org [6]. Conceptually speaking, each decision tree generated by this algorithm could be thought of as an individual player who decides whether a character is good or not, given their features. I chose to use a large number of estimators (number of estimators = 1000), and did not exclude any number of features, since the feature data was already limited as mentioned. Other regression methods such as Ridge, Extra Trees, and Gradient Boosting were all tested, but yielded significantly worse results. As expected, these models were heavily overfitted due to the data's small size. Averaging the predictions of all the models together yielded somewhat better results, but the Random Forest still performed best. A Random Forest model was also used as a classifier once the problem was redefined (see 4.2).

4 Results

4.1 Regression

It was found that the models generated by the Random Forest regressor were not consistent between different training data sets. Some models gave negative or virtually 0 R-Squared scores, while others yielded much better results. The maximum score of 0.27 was achieved by the second generated training set in data set B, or "B2." Comparison between the predictions of this model and actual target values are summarized below in Figure 3.

Rating Predictions for Test Set 'B2'						
ID	actual		predicted		difference	
	character	rating	rating	point diff	% diff	
7	Pikachu	4.30	3.14	-1.16	-26.95	
10	Greninja	3.94	2.93	-1.01	-25.56	
12	Wario	3.80	2.79	-1.01	-26.52	
13	Ike	3.80	2.70	-1.10	-29.01	
17	Ivysaur	3.47	2.53	-0.94	-27.05	
20	R.O.B.	3.33	2.26	-1.07	-32.03	
22	Marth	3.25	3.49	0.24	7.39	
26	Ness	2.92	2.09	-0.83	-28.30	
32	Simon/Richt	2.41	2.02	-0.39	-16.07	
33	Bowser	2.25	1.95	-0.30	-13.37	
40	Samus/Dark	1.96	1.84	-0.12	-5.92	
54	Ken	1.55	1.97	0.43	27.48	
55	Bayonetta	1.52	2.42	0.90	59.73	
60	Dr. Mario	1.39	2.28	0.89	63.79	
61	Robin	1.27	2.04	0.76	59.95	
66	Jigglypuff	0.95	2.59	1.63	171.29	
71	Kirby	0.52	2.42	1.91	369.59	

Figure 3: Model Predictions for Numeric Rating Values

There are a few different implications behind these results. Firstly, the large disparity between scores for models built using different data implies that we could find some optimal training set. So in theory, we could go through every possible combination of training samples and find the best result; however, this would not be practical even with such small data. For example, say we wanted

to look at all training sets with 50 samples; this would require us to analyze 1791242627540058576 different possible combinations. Even if we took training sets with 62 samples, we'd still have to look at 536211932256 different combinations. Furthermore, there is nothing noticeably similar between training sets that generated better models, so it would be difficult to narrow down which combinations we would want to test.

Additionally, these disparities are not necessarily explained only by the training data. Because the test sets were so small (22 samples at maximum), it is possible that the models were all similarly bad despite their scores, but the ones with the highest scores just happened to better predict the target for that particular test set (and not very well anyway). Clearly, it is very difficult to identify specific problems with such small data.

Despite these limitations, it was noted that although the model generated from B2 poorly predicted exact target values, it largely preserved the general order of the characters in the test set relative to each other, i.e. character 1 was still predicted to be rated higher than character 2, character 2 was higher than character 3, etc. Figure 4 below illustrates this preserved order: Pikachu > Greninja > Wario > Ike > Ivysaur > Bayonetta > Dr. Mario > Robin. These results gave me the idea to change the problem to classification, where each character was given a "tier" class, A through E.

Rating Order Predictions for Test Set 'B2'			
actual		predicted	
character	rating	character	rating
Pikachu	4.30	Marth	3.48916237
Greninja	3.94	Pikachu	3.14418426
Wario	3.80	Greninja	2.93492858
Ike	3.80	Wario	2.79387247
Ivysaur	3.47	Ike	2.69799972
R.O.B.	3.33	Jigglypuff	2.589465
Marth	3.25	Ivysaur	2.53355449
Ness	2.92	Kirby	2.42381977
Simon/Richt	2.41	Bayonetta	2.42017073
Bowser	2.25	Dr. Mario	2.28042513
Samus/Dark	1.96	R.O.B.	2.26176653
Ken	1.55	Ness	2.0907552
Bayonetta	1.52	Robin	2.03584451
Dr. Mario	1.39	Simon/Richt	2.02250993
Robin	1.27	Ken	1.97293259
Jigglypuff	0.95	Bowser	1.95131773
Kirby	0.52	Samus/Dark	1.84367914

Figure 4: Model Predictions for Numeric Rating Values

4.2 Classification

The original target data was edited to create two different data sets: one where tier was based on numeric rating value (y2), and another where tiers contained almost equal numbers of characters, regardless of rating (y3). As before, Random Forest was used to build the models. The maximum mean accuracy for the classifier generated from the B2 data was 0.41, a stark improvement from the regressor. The same data was also used to build a K-Nearest Neighbors classifier (with k = 1) which yielded similar results, summarized below in Figure 5.

Class/Tier Predictions for Test Set 'B2'						
	actual		y2		y3	
character	y2	y3	rand forest	k-neighbors	rand forest	k-neighbors
Pikachu	A	A	D	B	A	A
Greninja	B	A	A	A	B	A
Wario	B	A	C	C	A	C
Ike	B	A	B	B	A	A
Ivysaur	B	B	D	B	D	B
R.O.B.	B	B	D	D	C	D
Marth	B	B	A	A	A	A
Ness	C	B	D	C	C	C
Simon/Richter	C	C	D	B	C	B
Bowser	C	C	D	E	D	E
Samus/Dark	D	C	D	B	D	B
Ken	D	D	D	D	E	E
Bayonetta	D	D	D	D	C	D
Dr. Mario	D	D	D	D	D	E
Robin	D	E	D	D	D	D
Jigglypuff	E	E	D	A	D	A
Kirby	E	E	D	C	D	B

Figure 5: Model Predictions, in Order of Predicted Rating

One interesting detail to note is the predictions for the character Wario. Three times out of four, Wario was classified as C tier, while his actual target values were A or B. This meant the classifiers generally predicted Wario to be worse than he is thought to be. This can easily be accounted for given domain knowledge, as one characteristic of Wario that pro players find most powerful is one of his special moves, which is not accounted for in the training data. Similarly, the character Jigglypuff was always predicted to be in a higher tier than pros rated it, as one of Jigglypuff's main weaknesses is its small hit boxes and large hurt boxes, both of which are characteristics not accounted for in the data. The implication of these results is that the model's flaws are mostly due to lack of feature data, as opposed to lack of samples. To further confirm this idea, it should be noted that models built from training data set A performed worse than those built from data set B, suggesting that the additional features were important for obtaining improved results, i.e. a greater number of relevant features yields better models.

Other misclassifications also reveal human bias in the target ratings. The character Marth, for example, was always misclassified as A tier, above his rating of B, suggesting that Marth is better than he is thought to be. This makes sense, as one of the best characters in the game, Lucina, is an almost exact copy of Marth, the only difference being that Marth has a mechanic where the tip of his sword deals more damage than the rest of the blade. Competitive players often deem Marth to be worse than Lucina, because his damage output is not as consistent. However, this might cause a player to rate Marth as worse than he actually is, since there is a better character choice available. Despite this, Marth is not necessarily a "bad" or unusable character. These results suggest that with a more complete data set, we could reveal more biases in how pros rate certain characters, and reveal some characters to be viable or even good, despite players perceiving them to be bad. This would allow players to make more informed decisions about character choice, as opposed to relying on peoples' opinions.

Feature importance was also considered for the best classifier. It was found that no single feature was very significant, as all features had importance values below 0.04 (4%). However, the features with virtually 0 importance were all among the extra proxy variables for recovery and hit/hurt boxes I added, summarized below (see Figure 6). This further supports the conclusion that more data is needed to create a more accurate model.

Feature Importance (orange if less than 0.01)											
attributes	air accel	0.0232	Jab	0.0103	Jab	0.0162	Jab	0.0240	recovery	top hitbox?	0.0023
	air speed	0.0269	F-Tilt	0.0110	F-Tilt	0.0220	F-Tilt	0.0117		invulnerabilit	0.0033
	max fall spe	0.0224	U-Tilt	0.0088	U-Tilt	0.0219	U-Tilt	0.0208		max vertical	0.0130
	fast fall spe	0.0246	D-Tilt	0.0097	D-Tilt	0.0114	D-Tilt	0.0121		horizontal m	0.0076
	initial dash	0.0260	Dash Attack	0.0108	Dash Attack	0.0173	Dash Attack	0.0173	other	tether?	0.0015
	run speed	0.0154	F-Smash	0.0190	F-Smash	0.0198	F-Smash	0.0136		projectile?	0.0025
	walk speed	0.0199	U-Smash	0.0139	U-Smash	0.0190	U-Smash	0.0086		reflector?	0.0040
	weight	0.0204	D-Smash	0.0118	D-Smash	0.0398	D-Smash	0.0107		counter?	0.0032
	full hop	0.0112	N-Air	0.0157	N-Air	0.0156	N-Air	0.0143		max jumps	0.0022
	short hop	0.0179	F-Air	0.0177	F-Air	0.0146	F-Air	0.0130		weapon?	0.0047
	air jump	0.0153	B-Air	0.0218	B-Air	0.0176	B-Air	0.0128			
			U-Air	0.0086	U-Air	0.0098	U-Air	0.0132			
			D-Air	0.0245	D-Air	0.0221	D-Air	0.0182			
startup			Grab	0.0089			Pummel	0.0035			
			Dash Grab	0.0134			Forward Thr	0.0164			
			Pivot Grab	0.0141			Back Throw	0.0204			
							Up Throw	0.0209			
							Down Throw	0.0142			
advantage											
damage											

Figure 6: Feature Importances for Best Classifier

It is also interesting to note that pummel damage was one of the least significant features. This makes sense given domain knowledge, as the damage dealt by this particular move is not very important in competitive play. These results begin to reveal some of the game's complexity, where a large number of character features qualify performance, and no one feature is heavily weighted against all others.

5 Conclusions & Next Steps

It was found that Random Forest, along with other models, generated poor regression models from the training data. However, it was noted that the best model largely preserved the order of character ratings, and so the problem was altered for classification. The classifier built from the same training data using the Random Forest algorithm was significantly more successful. Furthermore, the results pointed towards clear problems with the training data, in particular a lack of key features not present in the data currently available. These results suggest that the particular data crucial for furthering this study would be hit and hurt box data, and a constructed summary of special moves.

The results of this study are promising, and suggest that more successful models can be built with further data collection, as opposed to increasing sample size, which is impossible given the limited number of characters. In addition, failure to generate successful regression models using the given target rating values, in addition to misclassification of certain characters, implies that the ratings themselves, based solely on pro player opinion, are not entirely accurate. This suggests that some characters are much better or worse than pros believe they are. With more data, players could discover that a character everyone believes is awful is actually usable or even very good, and could be used at high-level play. Finally, this study demonstrates how thoughtful, extensive feature engineering can be used to produce workable results, despite small data.

Acknowledgments

I would like to thank Professor Engelhardt and the TAs, for their teaching and support throughout the semester. I learned so much in this course, and the work, though challenging, was incredibly rewarding. I would also like to thank my good friend, Hari Rajagopalan, for sharing his more extensive knowledge of Smash Ultimate, and advising me on how to best summarize the large amount of complex data I had to collect and organize for this project.

References

- [1] Beefy Smash Doods. Who Can Climb THE WALL In Smash Ultimate? YouTube, YouTube, 3 May 2019, www.youtube.com/watch?v=Tg8oXDob960.
- [2] Branniglenn, Zapp. Smash Ultimate Frame Data Patch 3.0 (Complete). Smashboards, 29 Apr. 2019, smashboards.com/threads/smash-ultimate-frame-data-patch-3-0-complete.465028/.

- [3] Brownlee, Jason. Discover Feature Engineering, How to Engineer Features and How to Get Good at It. Machine Learning Mastery, 26 Sept. 2018, machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it
- [4] inktivate. April Ultimate Tier List - a Combination of 18 Different Pros' Lists from the Past 50 Days. Reddit, Apr. 2019, www.reddit.com/r/smashbros/comments/bagz79/april_ultimate_tier_list_a_combination_of_18/.
- [5] Kurogane. Smash Ultimate Frame Data Repository. Kurogane Hammer, 2019, kuroganehammer.com/.
- [6] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011; 12; p. 2825-2830. <https://scikit-learn.org/stable/>.
- [7] Sirlin, David. Balancing Multiplayer Competitive Games. GDC Vault, UBM Technology Group, 2009, www.gdcvault.com/play/1570/Balancing-Multiplayer-Competitive.
- [8] SmashWiki. SmashWiki, 2019, www.ssbwiki.com/.
- [9] Wonkey. Harada on Revealing Frame Data in TekkenGames. Avoiding The Puddle, 2014, www.avoidingthepuddle.com/news/2014/2/4/harada-on-revealing-frame-data-in-tekken-games.html.
- [10] . Untitled Animation - All Characters High Jump Championship. Facebook, Facebook, Apr. 2019, www.facebook.com/UntitledAnimation/videos/348093746052624/.

A Data Collection/Cleaning Notes

A.1 General Notes

- all data was taken from Patch 2.0 of Smash Ultimate; this includes ratings given by professional players for target values (considered only tier lists released between Patches 2.0 and 3.0)
- frame data was taken for the following moves: jab, smash, aerial, tilt, grab, throw, pummel (no specials)
 - properties noted: base damage, startup, advantage
 - damage for moves with multiple hits was taken as the sum of damage dealt by each individual hit
 - for rapid jabs, five hits and the jab finisher were assumed; another option would be to separate jab into separate features
 - for moves with multiple hit boxes, damage was taken from only the more consistently used hit box (usually the first one listed); another option would be to average the damage dealt by each hit box; the most accurate option would be to weigh damage from each hit box by the area of the hit box, but we lack this data

A.2 Specific Character Properties

- data for Rosalina and Olimar was omitted entirely; difficult to measure which Pikmin are being used/whether Luma or Rosalina or both are delivering hits, as this varies circumstantially
- Shulk was considered without Monado Arts
- Lucario was considered at 65%
- damage for Donkey Kong's cargo throw was averaged

A.3 Missing Data

- new DLC characters from Smash 4, and new characters in Smash Ultimate, do not have frame data available from Patch 2.0; this data was taken from Patch 3.0 (not too many changes between these patches affecting these characters, except Corrin); this includes Ryu/Ken, Cloud, Corrin, Bayonetta, Inkling, Ridley, Simon/Richter, King K. Rool, Isabelle, Incineroar, and Piranha Plant
- Ness' D-air was not in the data, so this was taken from Smash 4 data from Kurogane Hammer [5]
- Ken's throw damage is taken from Ryu's data (Ken's is missing for some reason)
- Kirby's dash attack has no advantage data, so a value of 0 was assumed
- vertical height for Piranha Plant was estimated from Beefy Smash Doods' video on who can climb the high wall (was seen to be between King Dedede and Pit/Dark Pit) [1]
 - the "Untitled Animation" video showing maximum vertical heights was also not used for Ivysaur and Lucario [10]; Ivysaur's distance was extended to include the tip of Vine Whip, and Lucario's height was recorded manually at 65%