Dynamic Generation of Statblocks for Pathfinder 2nd Edition through Machine Learning Models

Vess

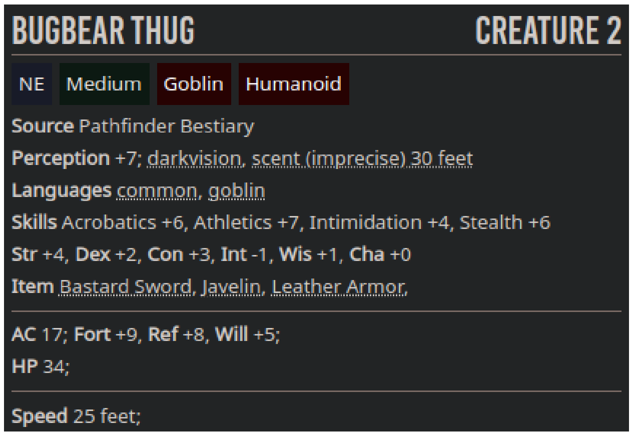
Thanks to Okami and Strat for editing.

April 2023

1 Problem Identification

Tabletop roleplaying games (TTRPGs) are an increasingly popular pastime for many board game and war game enthusiasts. TTRPGs are generally played through two roles– the typically single Game Master (GM) / Dungeon Master (DM), and a group of two players or more. The GM utilizes a TTRPG system with a "system book", such as Dungeons & Dragons or Pathfinder, which acts as a rules reference to run the players through homemade or previously published content. The GM performs rules adjudication guided by these system books for various actions the players wish to perform in the shared narrative game world. These TTRPGs are usually experienced through two primary modes of play, roleplaying and combat.

In the combat mode of play, GMs rely on premade and hopefully balanced "statblocks" for various monsters, creatures, and enemies, that the players can typically encounter and fight in the course of both modes of gameplay. Statblocks consist of numerical values that are abstract representations of an enemy's features, such as strength, dexterity, constitution, and so on, with each of these statistics providing mechanical system benefits. While GMs can normally rely on a wealth of these premade statblocks for many enemy types, there are various statblock concepts that are still yet unrepresented by this premade set. Although GMs can naturally sit down and design their own enemy statblocks per PF2e rules [1], this can time-consuming to design properly relative to the skill of experienced game designers who have a stronger familiarity with their domain specific systems.


BUGBEAR THUG CREATURE 2

NE Medium Goblin Humanoid

Source Pathfinder Bestiary
Perception +7; darkvision, scent (imprecise) 30 feet
Languages common, goblin
Skills Acrobatics +6, Athletics +7, Intimidation +4, Stealth +6
Str +4, **Dex** +2, **Con** +3, **Int** -1, **Wis** +1, **Cha** +0
Item Bastard Sword, Javelin, Leather Armor,

AC 17; **Fort** +9, **Ref** +8, **Will** +5;
HP 34;

Speed 25 feet;

Figure 1: An Example of a PF2e Statblock

This is where and why a consideration for ML (Machine Learning) seemed to be an obvious answer. The bulk of premade statblocks mainly consist of numerical values and ordinal types. Given that there exists this preestablished, extensive, and balanced set of statblocks, there is a prime opportunity to train ML models on them and attempt the automation subsequent generation of similar statblocks for GM usage on an as-needed basis. This would save time for TTRPG gamers, provide unique statblocks to make use of, and also present an opportunity for further understanding of how enemy features interrelate. This subject was the focus of my study.

In section 2, I provide a recap on the current published literature concerning this problem domain. Section 3 discusses various technical concepts and technology that I utilized for the project. Section 4 was my methodology and process for the project. Section 5 is a presentation of various results achieved, both interesting and disappointing. Section 6 is directed towards possible ideas on how to iterate and improve on my work in future endeavors.

2 Literature and History

The utilization of ML and artificial intelligence within the scope of TTRPGs like Dungeons and Dragons (DnD) and Pathfinder (PF) remains relatively unresearched. However, with the recent release of ChatGPT [2] there has been a spur of interest in the application of adaptive algorithms towards these style of games. ChatGPT users were quick to teach the AI slimmed down rules of prototypical TTRPGs and have the AI function as a sort of robot-DM for them [3]. ChatGPT was able to pay attention to the mechanics relayed to it, creatively interpret the input, and provide much of the same experience as that of the job of a human GM with only minor deviations. This community astonishment seemed not to have escaped the eye of the owner of the DnD property, WoTC (Wizards of the Coast). Leaked internal plans for the company, later corroborated by internal sources within the company, showed that WoTC had interest in the inclusion of "AI-DMs" [4] for solo players or other player groups that lacked one. Thereby the potential of both untapped interest and the opportunity for more targeted research seems enticing, warranted, and necessary.

Apart from ChatGPT, which pointedly is not specifically trained for the task of AI-DMing and thus falls short in many regards of a fully realized vision of it, there have been a few scant recent pushes on this front in terms of formal research. Researchers have looked not to generate the statblocks of TTRPGs, but of players [5], taking in a biography or description and deriving the rest of a character sheet for them from said text. Dynamically their model was able to provide some prediction of a player character's preferred race, class, and "attribute scores" [5] which is the DnD system specific term for the aforementioned stats like strength, dexterity, constitution, etc. While some have curated a particular interest in DnD statistical relations [6, 7] most informally published research has primarily only concerned itself with data mining for relations in DnD stats [8]. These were perhaps the closest kinds of research to mine, but lacked my interest of the next step– which is on-demand dynamic generation with ML models.

For an understanding of why DnD research will apply to my PF research, a historical clarification must be made. TTRPG system books are updated and released in iterative fashion, known as editions. DnD has to date received five such editions, with DnD 5e being the current edition. In the transition between editions known as DnD 3.5e and DnD 4e, the once permissive licensing of DnD would receive changes that caused a schism in the community [9]. PF was a spinoff system built on the foundations of DnD 3.5e under its more permissive licensing. Pathfinder would eventually receive its own updated version, colloquially referred to as PF2e. While PF and consequentially PF2e have received various changes that distinguish them from DnD 3.5e, much of the same "DnD skeleton" still exists within PF properties, and thus DnD research remains applicable and useful to mine.

In the same vein of domain-specific research to DnD being that it is one of the most widely known and popular systems, and thus is one of the few TTRPGs to receive any academic attention, there have been a few forays into answering questions about what it would be like to apply ML concepts towards the subject. Some researchers have approached the topic and astutely pointed out DnD and other TTRPGs' apt usage for ML model study [10, 11]. Given that DnD and TTRPGs in general incorporate both structured math in the form of statistical allegories, and structured play in the form of dice, applied statistics, and some bounded randomness, as well as a high propensity for large natural language datasets, it's no surprise that the surprisingly little research I could find concerned attempts to teach AIs how to generate narratives and play DnD.

A less stat focused research paper concerned itself with attempts to learn, interpret, and then produce DnD-like narratives through the data mining of TTRPG transcripts [12]. Such interest was not alone as later research concerned itself with full generation of typical DnD game turns [13]. They attempted to model both the roleplaying and combat modes of DnD, incorporating similarly large transcript datasets [13]. While the results were not as widely notable as ChatGPT's success has been in this relatively unconsidered field of study, the progress and the lessons learned cannot be discounted. While the interest of my research was primarily concerned with a specific domain of TTRPG data mining and dynamic generation (Pathfinder statblocks), the work towards predicting average DnD character statistics [5], previous work of my own with DnD statblock statistics [14], and the full integration and parsing of DnD combat (the mechanical application of DnD statblocks) [13], proved to be useful references.

3 Concepts and Technology

Due to previous experience in a Data Mining class where I employed similar techniques regarding DnD statblock parsing and mining [14], there were not that many concepts in completing the project that were unfamiliar to me. However, the project was not completed in full without a furthering of my understanding of various ML terminology that was previously unknown to me, and how the technology behind them functioned. For a more clear understanding of my process and final implementation, three things will be clarified. First, what Multi-Ouput is and how it functions in relation to ML models. Second, the difference between Classifier and Regressor ML models. And third, a clarification on the metrics utilized, what a voting model does, and the terms used by the PF2e data.

3.1 Multi-Ouput Models

Early on in my research and the beginning of the development of my solution implementation, I experienced difficulties in getting my models to predict multiple targets at the same time. "Targets" refers to the unknown variables the user wishes to predict at the end of model training. Previous education on ML primarily concerned only the addressment of singular target predictions by ML models. Such as identifying what type of character a written letter was– the singular target being the character type, like A, B, C, etc. However, what if I gave you the height of someone and then asked you what their probable weight and age were? This is where my models would fall short– and online research for keywords would always simply return guides on Synthetic Data generation– aka how to create fake data for use with ML models. It would be late into my project when I finally found the right terminology to use to get what I was looking for, which was multi-output. Multi-output simply means that a model is separately trained on finding each target that you're looking for from your inputs, and then chains these models together in such a way that they can predict for multiple outputs in unison [15].

3.2 Classifier and Regressor Models

Among ML algorithms, regressor and classifiers are two notable types that help with making predictions on how new and previously unseen data might be interpreted and categorized. In the strictest sense, classifier models are used in the classification of discrete categories such as letter recognition, computer vision, and other similar problems concerning identification. Regressor models are used when the problem has values that are continuous and interrelated; like housing prices, the stock market, and weather forecasts– where all data can sequentially be used to gain a further understanding of a trend [16]. This however begs an interesting question– which model then consequentially better fits the problem space of statblocks? There are definitely distinct categories, such as a statblock's level, but also deeper correlations between how values trend– such as a statblock's Strength bonus and its Athletics skill bonus. This was simply addressed in my methodology by employing both model types and then comparing the results.

3.3 Technical Details

Given that I would end up utilizing both classifier and regressor models, two different metrics of evaluation were used for each model's performance based on their type. Classifiers were evaluated based on their accuracy statistic, which is the number of accurate predictions made by the model on unseen data divided by the total number of predictions, correct and incorrect. In contrast while classifiers make boolean predictions that an input either is or isn't in some category, regressors shoot for decimal point values that represent how well an independent variable influences a given dependent variable. This scoring method is known as an R-Squared score.

In addition to classifier and regressor models, I would employ a voting model at the end for both model groups. A voting model is somewhat self-explanatory– you take a collection of models, give them an input, and then have them take a majority vote on what they believe the data should be categorized as. There were technically two distinct voting model types employed in my project, one specifically tailored for classifiers, and one for regressors.

By the end of the project there were 32 distinct data types identified (excluding name) and kept for their usefulness in model training for each statblock. Datapoints such as traits, languages, resistances, and attacks, were also excluded due to their lack of usefulness for statistical analysis or difficulty in parsing and defining them in numerical terms. For an understanding of the chosen values and how they might possibly interrelate in such a fashion a small explanation of each is provided. Additionally the term "modifier" should be made clear. TTRPGs often employ a system of reducing larger numbers to smaller more manageable numbers that tier off the progress and development of certain facets of a character. For example while the specifics of modifiers differ between TTRPGs, in PF2e a character (or statblock) with 10-11 strength will have a +0 strength modifier. If they have 12-13 strength then they will have a +1 strength modifier. The modifier is increased for every two given ranks of an ability score (what DnD refers to as attribute scores).

The PF2e data identified as useful is as follows,

1. A statblock's level is from 1-20, and represents the "difficulty" of what fighting the statblock would be like. For example a party of four level 2 adventurers is expected to be capable of fighting a level 2 statblock.

2. Then there is size, which can be tiny, small, medium, large, huge, or gargantuan. A statblock's size indirectly correlates with AC (Armor Class), HP (Health Points), and ability scores.

3. Then there is a ranking of a statblock's lawfulness from chaotic, neutral, to lawful. Lawfulness generally correlates with no other statistic.

4. And a ranking of a statblock's morality from evil, neutral, to good. Morality also generally correlates with no other statistic.

5. Next every statblock has an AC representing how difficult the statblock is to hit for any given attack. Typically a statblock's AC indirectly correlates with its dexterity modifier or level.

6. Then there are HP, which represent how much damage a statblock can take before it dies. HP directly correlates with a statblock's constitution modifier, and indirectly with its level.

7. There is a perception bonus to how observant a statblock is to its surroundings, and directly correlates with the statblock's wisdom modifier.

8. – 10. A statblock has fortitude, reflex, and willpower saving throw modifiers, which directly correlate with a statblock's constitution, dexterity, and wisdom modifiers, respectively.

11. – 16. A statblock has modifiers for their strength, dexterity, constitution, intelligence, wisdom, and charisma. These indirectly correlate with level.

17. – 32. A statblock may or may not have various skill modifiers– where each modifier directly correlates with an associated ability score modifier. For example, strength and athletics. A skill modifier of 0 was assumed for all skills not present in a given statblock, per standard PF2e rules [17].

4 Methodology and Process

My methodology remained similar to the one that I employed in a previous approach to DnD statblock data mining [14], albeit with some key differences. Where I was able to simply download a premade DnD statblock data collection before, due to the difference in popularity between DnD and PF2e there existed no such premade repertoire for my purposes for PF2e. Thankfully however due to PF2e's permissive community licensing usage policy [18], all of their published commercial content was online and available for noncommercial personal use. Thusly the first step of the process, and nearly 60% of the total work, was scraping, parsing, and cleaning the data from the web. Then the following 40% of the work was spent on training my models en masse on the generated dataset before finally figuring out how to get them to be multi-output.

4.1 Scraping, Parsing, and Cleaning

My first step was primarily concerned with collecting a prospective list of possible candidate websites to scrape from. One such website was eventually narrowed down to from a few others, and was considered to be the best possible candidate given its clean and accessible formatting [19]. Amusingly the website makes usage of the same PF2e policy as I was, and also gathered its database from a different online source as well. The website presents each statblock from different unique HTML files that contain the respective information and statistics for said statblock. This made scraping the website a viable opportunity and relatively easy barrier to address for building out the necessary data collection I would require for my project. It should be noted that despite it being legal to scrape public websites, I took steps to ensure that my script only scraped the website once and then made backups– additionally breaking each step of the process into discrete parts that did not repeat once they were finished.

Due to Python 3's aptness with problems like data scraping and parsing, and ML, it was the language I picked for this project. Utilizing a Python script that I wrote, I then targeted and scraped $4,923$ unique PF2e statblocks. During this intake process, I consequentially identified several possible problematic statblocks that might have weighted my models if I had left them included. These were AV, BB, Class, Elite, Improvised, Generic, PFS, and Spellcaster, typed statblocks. Additional "leveled" statblocks were also excluded. The differences between each are inconsequential in the recounting of this methodology, but essentially, these were statblocks with small variations between them that were present in the dataset due to different publishing runs by Pazio (PF2e's owner) for PF2e that resulted in duplicated statistics. Two additional statblock types were identified: Petitioners and Soulbound Dolls, which were also generally repeats. This lead to the exclusion of $1,229$ statblocks, leaving me with $3,694$ prospective statblocks. From these statblocks, a further issue was identified, where there were statblocks for "hazards" included– which functioned wholly differently than regular monster / creature / enemy statblocks did. Thusly 540 of these hazard statblocks were then excluded, leading to the final count of $3,154$ statblocks, which at 32 distinct datapoints left me with $100,928$ values to train my ML models on.

Finally my script dug into the scraped HTML of each statblock and utilized regex matches to locate the

32 identified datapoints. Each datapoint was then extracted and exported to a CSV file. During this extraction process, a further issue was identified. Some statblocks, less than a dozen, had 404'd on the original scraped website. I performed due diligence of reaching out to notify the web admin, and these statbocks were just retroactively added to the previous statblock filters, before the process was repeated from that point onwards again. Additionally some malformed and sometimes incorrect statblock statistics were identified– and these were either also simply filtered or replaced with their obvious fixed values. My processing script also separated out a statblock's alignment into its separate lawfulness and morality counterparts, as well as replacing absent skill modifier bonuses with zeroes. As a final touch, my script mapped each statblock's size, lawfulness, and morality values, onto ordinal values instead of their previous string types– because ML models generally do not accept strings as valid values.

4.2 Data Mining and Generation

My usage of Python continued, as it excelled in the problem domain of ML due to the stable, mature, and extensive existing available ML and data mining libraries. Of the libraries, Scikit [20] was the most favorable for my conditions, and is one that I already had previous experience working with [14]. Scikit provides for a variety of regressor, classifier, multi-output, and voting models.

The first step of the main interest of this project was to import and prepare my PF2e statblock data. In the effort of working towards the completion of my final goal of dynamic statblock generation, I setup an ease of use mock reference statblock that allows the executor of the program to define which mock values are provided to the models, and will dynamically split my statblock data accordingly. Based on which values are not provided to the model, the program separates out all relevant datapoints that match, and places them into a target set (the data to predict). The program then keeps the rest of the data as a reference set (the data to work off of). A per execution random percentage of the target and reference set collectively are then separated out and utilized as unseen data through which the metrics of the models are evaluated.

Then from Scikit, 51 regressor models were imported. 24 were soon excluded due to their unfavorable conditions, such as a lack of support for multi-output, an inability to run without more data entries, a lack of speed and efficiency when compared to other models, or consistently horrible scoring on my unseen data. This left me with 27 regressor models. From Scikit I additionally imported 37 classifier models. Of these, 30 models were excluded for similar reasons to the excluded regressor models. This left me with 7 classifier models. My 27 regressors and 7 classifiers were then wrapped in a multi-output model before being trained in sequential order on the reference and target sets.

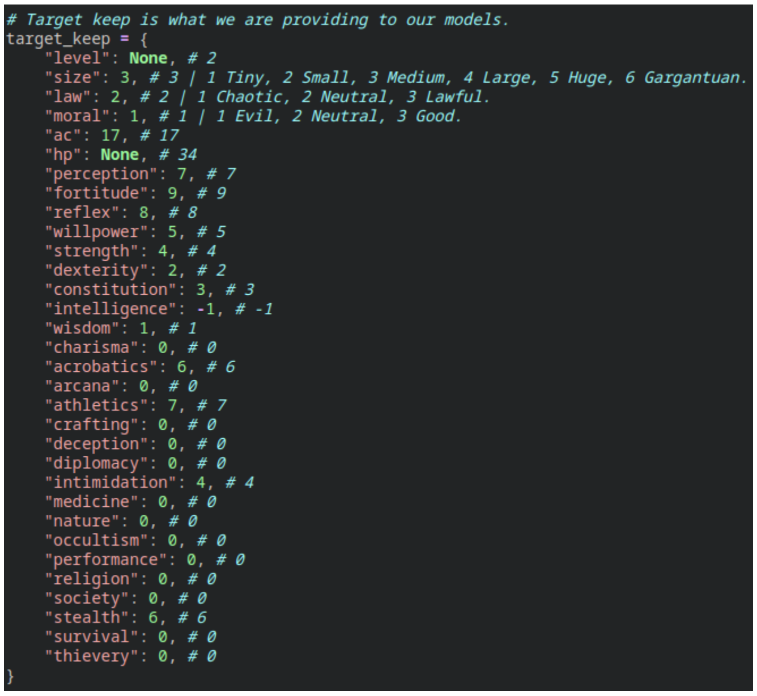
Finally after training each model was evaluated against the unseen set, and ordered according to their score. From this abstract model leaderboard, the top three of the regressor and classifier models are selected and then placed into a voting model. I then once again train both the regressor and classifier voting models on the reference and target sets, before one last evaluation of their score against the unseen set. After this I finally used these voting models to fill in the blanks in the original mock reference statblock. Thusly, dynamic generation of statblocks for Pathfinder 2nd Edition through machine learning models was achieved. But how decent were the results of the models?

5 Result Presentation

The results of my models were at some times surprising, but mostly downright disappointing. In comparison to my previous work [14] however the models still performed significantly better. It is my belief that the reason for this relates to the difference in the sizes of the datasets utilized. My DnD work used only 763 statblocks, compared to my now $3,154$ PF2e statblocks. While it should be noted that models generally perform better when trained on quality data over mass quantities of it– there is definitely a minimum threshold of the amount of data required to produce half-decent accuracy.

For the results of my project, two mock reference statblocks will be compared and discussed. One where the level and HP are not provided, and one where they are the only values provided. In the first instance the models are being asked to predict only two unique datapoints, and the in the latter as much as 30. Each example will include a variety of pictures. The first will be the provided mock reference statblock that asks the model to predict certain values. Then pictures of the results of the regressor and classifier voting models, which include the names of the top three models along with their score, final combined score of the voting model, and the total time taken for training the voting model. At the end the final predicted values by the two voting models are given.

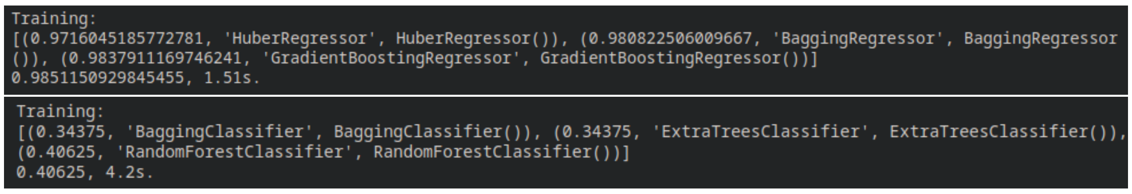
5.1 30 Datapoints Example



```
# Target keep is what we are providing to our models.
target_keep = {
    "level": None, # 2
    "size": 3, # 3 | 1 Tiny, 2 Small, 3 Medium, 4 Large, 5 Huge, 6 Gargantuan.
    "law": 2, # 2 | 1 Chaotic, 2 Neutral, 3 Lawful.
    "moral": 1, # 1 | 1 Evil, 2 Neutral, 3 Good.
    "ac": 17, # 17
    "hp": None, # 34
    "perception": 7, # 7
    "fortitude": 9, # 9
    "reflex": 8, # 8
    "willpower": 5, # 5
    "strength": 4, # 4
    "dexterity": 2, # 2
    "constitution": 3, # 3
    "intelligence": -1, # -1
    "wisdom": 1, # 1
    "charisma": 0, # 0
    "acrobatics": 6, # 6
    "arcana": 0, # 0
    "athletics": 7, # 7
    "crafting": 0, # 0
    "deception": 0, # 0
    "diplomacy": 0, # 0
    "intimidation": 4, # 4
    "medicine": 0, # 0
    "nature": 0, # 0
    "occultism": 0, # 0
    "performance": 0, # 0
    "religion": 0, # 0
    "society": 0, # 0
    "stealth": 6, # 6
    "survival": 0, # 0
    "thievery": 0, # 0
}
```

Figure 2: Mock Reference Statblock with 30 Values Given

Seen above are the given 30 datapoints. Level and HP are not given to the models, as denoted ”None”. Through this the script will separate out the target sets accordingly. The expected values are left as comments next to each field. For level this is 2, and for HP this is 34.

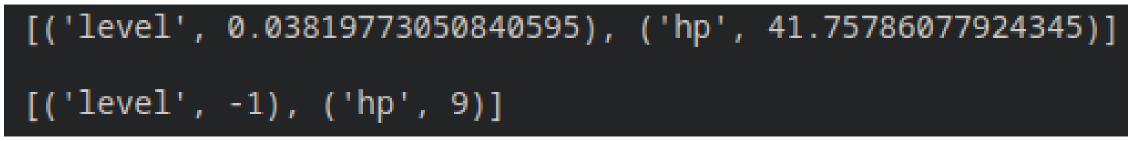


```
Training:
[(0.9716045185772781, 'HuberRegressor', HuberRegressor()), (0.980822506009667, 'BaggingRegressor', BaggingRegressor
()), (0.9837911169746241, 'GradientBoostingRegressor', GradientBoostingRegressor())]
0.9851150929845455, 1.51s.
```

```
Training:
[(0.34375, 'BaggingClassifier', BaggingClassifier()), (0.34375, 'ExtraTreesClassifier', ExtraTreesClassifier()),
(0.40625, 'RandomForestClassifier', RandomForestClassifier())]
0.40625, 4.2s.
```

Figure 3: Results of the Regressor and Classifier Voting Models

Seen above are the top three regressor and classifier models, put into two voting models, trained, and then evaluated– listing a score and execution time at the bottom. The scores of the six models can been seen listed as well.



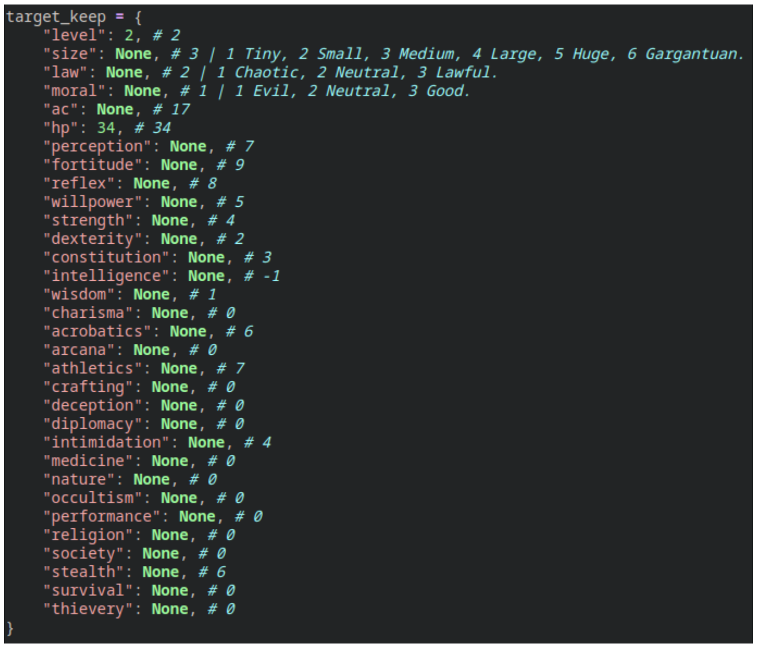
```
[('level', 0.03819773050840595), ('hp', 41.75786077924345)]

[('level', -1), ('hp', 9)]
```

Figure 4: Final Predicted Values of the Regressor and Classifier Voting Models

Seen above are the predictions for the level and hp from the regressor and classifier voting models.

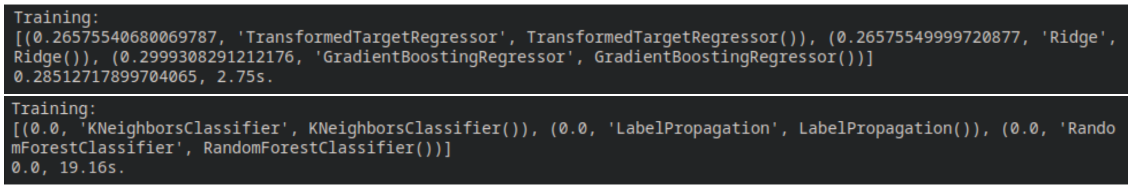
5.2 2 Datapoints Example



```
target_keep = {
    "level": 2, # 2
    "size": None, # 3 | 1 Tiny, 2 Small, 3 Medium, 4 Large, 5 Huge, 6 Gargantuan.
    "law": None, # 2 | 1 Chaotic, 2 Neutral, 3 Lawful.
    "moral": None, # 1 | 1 Evil, 2 Neutral, 3 Good.
    "ac": None, # 17
    "hp": 34, # 34
    "perception": None, # 7
    "fortitude": None, # 9
    "reflex": None, # 8
    "willpower": None, # 5
    "strength": None, # 4
    "dexterity": None, # 2
    "constitution": None, # 3
    "intelligence": None, # -1
    "wisdom": None, # 1
    "charisma": None, # 0
    "acrobatics": None, # 6
    "arcana": None, # 0
    "athletics": None, # 7
    "crafting": None, # 0
    "deception": None, # 0
    "diplomacy": None, # 0
    "intimidation": None, # 4
    "medicine": None, # 0
    "nature": None, # 0
    "occultism": None, # 0
    "performance": None, # 0
    "religion": None, # 0
    "society": None, # 0
    "stealth": None, # 6
    "survival": None, # 0
    "thievery": None, # 0
}
```

Figure 5: Mock Reference Statblock with 2 Values Given

Seen above are the given 2 datapoints. 30 datapoints are not given to the models, as denoted ”None”. Through this the script will separate out the target sets accordingly. The expected values are left as comments next to each field.

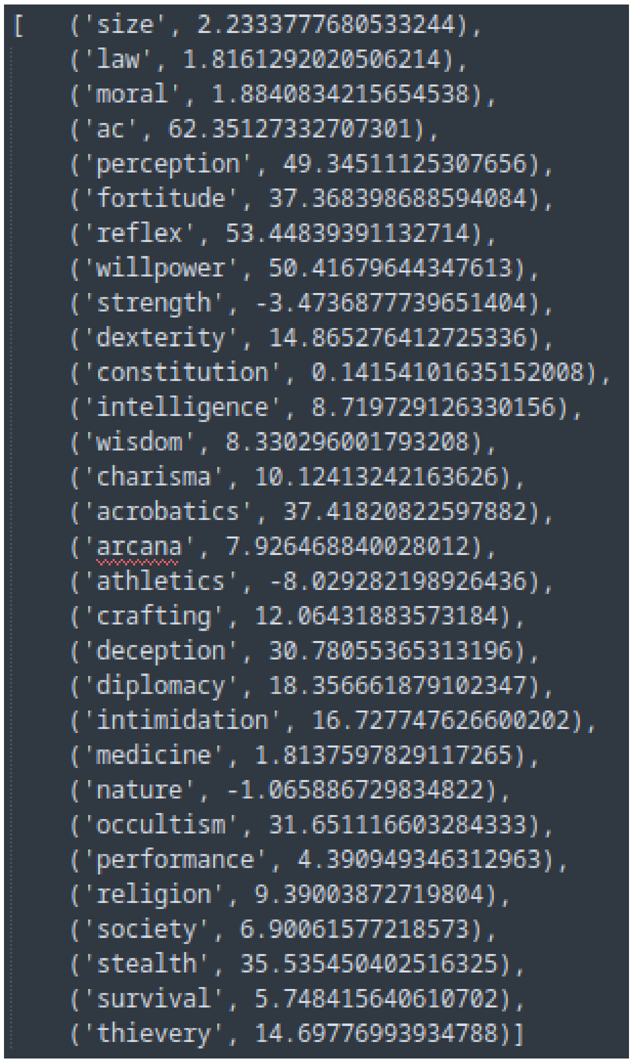


```
Training:
[(0.26575540680069787, 'TransformedTargetRegressor', TransformedTargetRegressor()), (0.26575549999720877, 'Ridge',
Ridge()), (0.2999308291212176, 'GradientBoostingRegressor', GradientBoostingRegressor())]
0.28512717899704065, 2.75s.
```

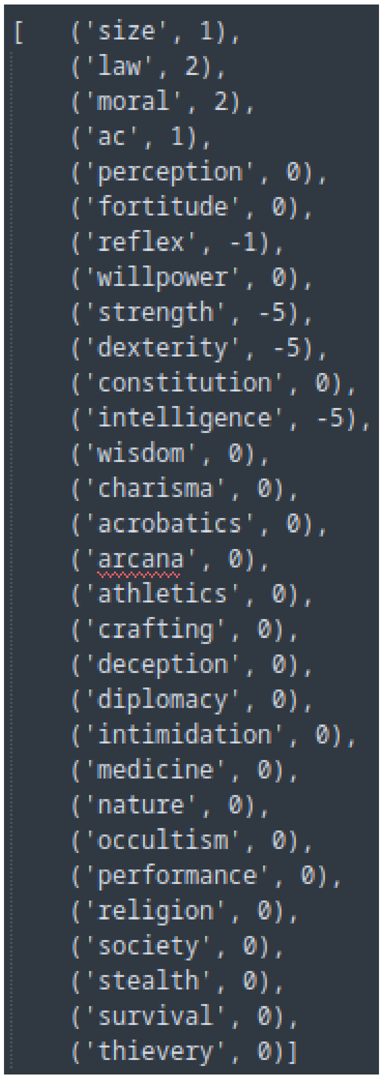
```
Training:
[(0.0, 'KNeighborsClassifier', KNeighborsClassifier()), (0.0, 'LabelPropagation', LabelPropagation()), (0.0, 'Rando
mForestClassifier', RandomForestClassifier())]
0.0, 19.16s.
```

Figure 6: Results of the Regressor and Classifier Voting Models

Seen above are the top three regressor and classifier models, put into two voting models, trained, and then evaluated– listing a score and execution time at the bottom. The scores of the six models can been seen listed as well.



```
[   ('size', 2.2333777680533244),
    ('law', 1.8161292020506214),
    ('moral', 1.8840834215654538),
    ('ac', 62.35127332707301),
    ('perception', 49.34511125307656),
    ('fortitude', 37.368398688594084),
    ('reflex', 53.44839391132714),
    ('willpower', 50.41679644347613),
    ('strength', -3.473687773965140 4),
    ('dexterity', 14.865276412725336),
    ('constitution', 0.14154101635152008),
    ('intelligence', 8.719729126330156),
    ('wisdom', 8.330296001793208),
    ('charisma', 10.12413242163626),
    ('acrobatics', 37.41820822597882),
    ('arcana', 7.926468840028012),
    ('athletics', -8.029282198926436),
    ('crafting', 12.06431883573184),
    ('deception', 30.78055365313196),
    ('diplomacy', 18.356661879102347),
    ('intimidation', 16.727747626600202),
    ('medicine', 1.8137597829117265),
    ('nature', -1.065886729834822),
    ('occultism', 31.651116603284333),
    ('performance', 4.390949346312963),
    ('religion', 9.39003872719804),
    ('society', 6.90061577218573),
    ('stealth', 35.535450402516325),
    ('survival', 5.748415640610702),
    ('thievery', 14.69776993934788)]
```



```
[   ('size', 1),
    ('law', 2),
    ('moral', 2),
    ('ac', 1),
    ('perception', 0),
    ('fortitude', 0),
    ('reflex', -1),
    ('willpower', 0),
    ('strength', -5),
    ('dexterity', -5),
    ('constitution', 0),
    ('intelligence', -5),
    ('wisdom', 0),
    ('charisma', 0),
    ('acrobatics', 0),
    ('arcana', 0),
    ('athletics', 0),
    ('crafting', 0),
    ('deception', 0),
    ('diplomacy', 0),
    ('intimidation', 0),
    ('medicine', 0),
    ('nature', 0),
    ('occultism', 0),
    ('performance', 0),
    ('religion', 0),
    ('society', 0),
    ('stealth', 0),
    ('survival', 0),
    ('thievery', 0)]
```

Figure 7: Final Predicted Values of the Regressor and Classifier Voting Models

Seen above are the predictions for the 30 datapoints from the regressor and classifier voting models.

5.3 Example Conclusions

The results are interesting, but admittedly fall short of what I was personally hoping for when I embarked on this project. Consider figures 1 – 3 of the 30 provided values statblock. The regressor voting model gave a level of ~ 0.03, and an HP of ~ 41.75. While the HP is in the range of what I was looking for, the level definitely is not. Whereas the classifier voting model gave a level of -1 and an HP of 9... So far they are leaving a lot to be desired in comparison to their expected values of 2 and 34. Next consider the second example of the two provided values statblock. The regressor voting model generally gave values way outside any reasonable range, but some of the datapoints are at least within believable limits. The classifier model also gave some workable values, but most of it is unusable for any real purpose. If the models had worked better, the given values would have been closer to the expected values.

In further tweaking and testing, this trend continued. Some values were believable, possibly even entirely within the projected range of what an average statblock would contain, but mostly... Garbage. Despite the improvement over my previous work there was still a lot left to be desired. To generate any useful statblock, one would have to essentially reroll on the data generation and stitch together believable values. Comparatively this is still more expedient than designing your own statblock from the ground up, but at the rate at which my models perform one could simply ask ChatGPT and receive something entirely more reasonable. Heartbreaking. All in all however, this project still proved as a useful learning exercise about the dangers of ML... It doesn't simply solve all your problems when you would like it to like a magic box.

6 Future Iteration

There are plenty of finer points for consideration in the possible future iteration and improvement on the work that I've done here. My program still currently remains as a script that can be tweaked to run by someone with some scripting familiarity. It would perhaps be more productive to utilize free and open source Python compilation tools such as PyInstaller [21] to package a transportable executable for easier access and use. Additionally both the classifier, regressor, and voting models, were left without fine-tuning of their available configurable options. There could definitely be a measurably significant performance benefit to fine-tune some of the models around the data. And as a final note it should be pointed out that while $3,154$ statblocks are a sizeable dataset, the possible incorporation of some of the statblocks I chose to leave out, future released statblocks, or fan generated statblocks, may prove beneficial to add to my dataset for future model training and testing. However it serves well to remember that if you put garbage data in–you'll get garbage data out. Quality over quantity!

The code can be found on my personal Github under an open source license to enable such possible future endeavors here:

References

[1] *Building Creatures*. URL: https://2e.aonprd.com/Rules.aspx?ID=995.

[2] OpenAI. *ChatGPT: Optimizing Language Models for Dialogue*. Feb. 2023. URL: https://openai.com/blog/chatgpt/.

[3] Sean Murray. *Dungeons & Dragons Player Gets AI Bot To DM, Works Surprisingly Well*. Dec. 2022. URL: https://www.thegamer.com/dungeons-dragons-chatgpt-ai-dm/.

[4] Eric Law. *Rumor: Dungeons and Dragons may introduce a subscription system for one D&D*. Jan. 2023. URL: https://gamerant.com/dungeons-dragons-subscription-system-one-dnd-rumor/.

[5] Joe MacInnes. "The D&D Sorting Hat: Predicting Dungeons and Dragons Characters from Textual Backstories". In: 2019.

[6] *5E monster manual on a business card*. May 2022. URL: https://www.blogofholding.com/?p=7338.

[7] Dan Keefe. *Dungeons & Data*. May 2020. URL: https://peritract.github.io/2020/05/14/dungeons-and-data/.

[8] Cyberscribe. *d20datascience: Data Science investigations into the mechanics of the world's Greatest role playing game*. Aug. 2018. URL: https://github.com/cyberscribe/d20datascience.

[9] Greg Tito Legacy Author. *The state of D&D: Present*. Dec. 2011. URL: https://www.escapistmagazine.com/the-state-of-dd-present/.

[10] Lara J. Martin, Srijan Sood, and Mark O. Riedl. "Dungeons and DQNs: Toward Reinforcement Learning Agents that Play Tabletop Roleplaying Games". In: *INT/WICED@AIIDE*. 2018.

[11] Matthew Guzdial et al. "Tabletop Roleplaying Games as Procedural Content Generators". In: *CoRR* abs/2007.06108 (2020). arXiv: 2007.06108. URL: https://arxiv.org/abs/2007.06108.

[12] Annie Louis and Charles Sutton. "Deep Dungeons and Dragons: Learning Character-Action Interactions from Role-Playing Game Transcripts". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 708–713. DOI: 10.18653/v1/N18-2111. URL: https://aclanthology.org/N18-2111.

[13] Chris Callison-Burch et al. *Dungeons and Dragons as a Dialog Challenge for Artificial Intelligence*. 2022. DOI: 10.48550/ARXIV.2210.07109. URL: https://arxiv.org/abs/2210.07109.

[14] Vess-Dev. *DnD-Statistics: Throwing 4 machine learning models at DND creature statblocks*. Apr. 2022. URL: https://github.com/vess-dev/DnD-Statistics.

[15] Willy Ngashu. *Multi-output classification with Machine Learning*. Jan. 2022. URL: https://www.section.io/engineering-education/multi-output-classification-with-machine-learning/.

[16] *Regression vs classification in Machine Learning - Javatpoint*. URL: https://www.javatpoint.com/regression-vs-classification-in-machine-learning.

[17] *Skill Checks and Skill DCs*. URL: https://2e.aonprd.com/Rules.aspx?ID=174.

[18] *Paizo Inc. Community Use Policy Applicable to Non-Commercial Activity*. Aug. 2020. URL: https://paizo.com/community/communityuse.

[19] Ashley Hemerik. URL: https://pathfinderdashboard.com/.

[20] *scikit-learn: Machine Learning in Python*. URL: https://scikit-learn.org/stable/.

[21] *Pyinstaller Manual*. URL: https://pyinstaller.org/en/stable.