Classifying Creature Combat Capability in Dungeons and Dragons 5th Edition

A Hunter College CSCI-795 Project Report
December 21, 2021

Team:
John Lee
Alex Washburn

Contents

GitHub `https://github.com/recursion-ninja/CSCI-795-ML`

Video `https://github.com/recursion-ninja/CSCI-795-ML/recording/demo.md`

Abstract

In this project, we explore an under-developed aspect of Dungeons and Dragons 5th Edition (D&D), the monster "Challenge Rating" system. The Challenge Rating (CR) is a measure of a monster's lethality against a party of *four* characters, each that a character level equal to the monster's CR. Unfortunately, the supplied CR values that have been published are more often than not poor estimates of monster lethality. In this project, we attempt to produce an substitute ranking of monster lethality, ordering them into tiers of either `[1, 5]`, `[1, 10]`, or `[1, 20]`. We hypothesize that the application of machine learning will produce a better indicator of monster lethality than the supplied CR score.

Team Members' Roles

JOHN LEE

- Retrofitted and patched D&D combat simulator script (too much effort).
- Build and evaluated KNN, Logistic Regression, and XGBoost models.
- Authored project report and constructed presentation slide deck.
- Built, tuned, and evaluated the following models:
  - KNN-Classifier
  - Logisitc Regression Classifier
  - XGBoost Classifier.

ALEX WASHBURN

- Collected and curated D&D monster dataset.
- Authored project report and composed demo video.
  - Built, tuned, and evaluated the following models:
  - Decision Tree Classifier
  - Naive Bayes Classifier
  - Multi-layer Perceptron Classifier
  - Random Forest Classifier
  - Suport Vector (One-vs-one) Classifier
  - XGBoost Classifier.

State-of-the-art

Some work has been done on applying ML to table top role playing games (TTRPGs) in the past [1, 2, 3, 4, 5]. Much of the work revolves around the more tractable problem of selecting appropriate ambient music choices for players to experience based on the current emotional tone of the game [6, 7, 8, 9]. However, the most popular TTRPG, Dungeons and Dragons (D&D) has been used as a difficult test-bed for ML experimentation [10]. This particular previous work, while quite notable, focused entirely on non-combat aspects of D&D, eschewing a core component and past time of D&D; resolving conflict via numerical simulation. In our project we will grapple with this numeric aspect of D&D, focusing on a small subset of the D&D combat system; quantifying the relative lethality of a monster. To the best of the author's knowledge, the proposed project will be the first serious attempt to apply machine learning to a numerical aspect of D&D. Consequently, there is no *known* previous on which to draw a comparison.

Approach

D&D Monster Data

We take the stat-block of each monster from the `5e.tools` database and match the monster with the Elo Ranking from `dndcombat.com`. Elo Ranks scores on `dndcombat.com` are continually updated. The Elo observations were taken on `2021-12-08`.

The raw data used for this project exists within the repository for reference:

- `data/5e.tools/*`
- `data/dndcombat.com/*`

Both the data from `5e.tools` and `dndcombat.com` were retrieved in JSON format. The data was parsed and curated using a custom tool written by the authors. The tool is named `curate-json` and is written in Haskell. In order to build the `curate-json` tool, the Haskell compiler GHC and build tool `cabal` are required.

The source code for the `curate-json` tool within the repository for reference:

- `curation/*`

Additionally, there is a `curate-dataset.sh` script in the root of the repository which, assuming both GHC and `cabal` are installed, will replicate the data curation used for this project. The `curate-dataset.sh` script will place the curated dataset in the `data` directory.

The curated data used for this project exists within the repository for reference:

- `data/dnd-5e-monsters.csv`

Feature Extraction & Selection

The curated `dnd-5e-monsters.csv` dataset has `1386` rows and `72` columns, constituting the project's initial observations and measurements, respectively. There are two leading textual columns, labeled `'Name'` and `'Type'` indexed `[0, 1]`. The subsequent `20` columns indexed `[2, 21]` are "continuous," integral-valued measurements of common D&D attributes. The final `'Elo Rank'` column indexed `71` is the label for supervised machine learning models. The next three trailing columns, labeled `Damage Tags`, `Spellcasting Tags` and `Trait Tags` indexed `[66, 70]` The remaining columns indexed `[22, 67]` are binary indicators for various monster attributes. A column summary of the initial data set is presented in Table 6.

For our analysis, we desire the `'Elo Rank'` to be represented as an integral "tier list." We extract the tier list feature by discretizing the 'Elo Rank' column via following procedure:

1. Normalize the data into normally distributed quantiles via the `QuantileTransformer`.
2. Decide on the number of tiers: `[1, 5]`, `[1, 10]`, or `[1, 20]`.
3. Scale the data to fit the tier range.
4. Remove outliers.
5. Round values to nearest integer value.

Table 2: The removal of outliers will shrink the dataset depending on tier list size

Tier List	Observations
`[1,  5]`	`1200`
`[1, 10]`	`1316`
`[1, 20]`	`1348`

The final feature extraction involves the last three columns, labeled `Damage Tags`, `Spellcasting Tags` and `Trait Tags` indexed `[63, 65]`. Each can be expanded to extract an additional features, totaling `63` features combined. This feature extraction nearly doubles the fully expanded dataset size, now totaling to `135` features. The fully extracted dataset is presented in Table 9.

After extracting all possible features we perform feature selection.

First, the textual columns, labeled `'Name'` and `'Type'` indexed `[0, 1]` are dropped from our analysis. These columns do not influence combat efficacy and are not convertible to a meaningful numeric representation. Their absence makes the machine learning process much smoother.

Second, we drop all features extracted from the 'Damage Tag' and 'Spell Tag' columns. These extracted features had essentially no bearing on our first explored model (Multinomial Naive Bayes), and hence we decided to omit them from all future models for efficiency reasons.

For the final component of our feature selection process, we perform a "decorrelation" pass through the feature set. If any pair of features have a correlation coefficient of `0.6` or higher, we drop one of the columns and use the other as a proxy. Consequently, we dropped the features in Table 10. After feature extraction *and* feature selection, our dataset was comprised of `96` features. See Table 12 for the final feature set we used for our machine learning models.

Datset partitioning

We take the prepared dataset and train multiple machine learning classifiers. We use 80% of the randomly permuted data as the training set and the remaining 20% as the test set. This partition data was stratified by the `'Elo Rank'` column to ensure that each tier is represented. Furthermore, we partition the training set again, using 80% as a learning set and the remaining 20% as the validation set. Model selection was performed on the training set; comprised of the "learning" and validation subsets.

Table 4: Distribution for partitioning dataset, stratified by `'Elo Rank'`

Set	Ratio
Test	20%
Train	80%
Learn	64%
Validate	16%

Model Specification

1. **Decision Tree:** Decision trees make extremely fast classifiers once constructed, but can be incredibly time consuming to build. We decided to try our luck with this model and see if an effective classifier could be built within a reasonable time frame. We might get a surprising result, biut not placing a lot of hope in this model.
2. **K Nearest Neighbors:** A very simple model with a theoretical bound on it's maximum inaccuracy. Because of our familiarity with this model from the extensive discussion in class and it's prominent presence in our first homework, we chose this as our initial classifier to get our bearing and some quick benchmarking numbers.
3. **Logistic Regression:** We read that the a logistic regression can be an effective and efficient model for multi-class output, which our tier list is. This model was included because we suspected that the features had some linear, but not polynomial, relationship(s). The logistic regression ought to capture and train well if linear relationships exists between the features and the tier list labels.

4. **Multinomial Naive Bayes:** Independent features are am important factor for the efficacy of Naive Bayes models. Because we decorrelated our dataset, we felt that the remaining correlations beneath the `0.6` threshold was small enough to not interfere with the model's performance. Naive Bayes models are supposed to train well on small number of observations. Our dataset is just above the `1000` observation threshold, so we had high hopes that this model would train well. This was our second model used to get quick benchmarking numbers.
5. **Multi-layer Perceptron:** We wanted to experiment with the concept of artificial neural nets. The inclusion of this model allowed us to to get some experience with an instance of the buzzword-worthy model. Given the great flexibility of ANNs, we expected very good performance from this model.
6. **Random Forest:** Given the unknown nature and limited domain knowledge that we could use to direct the machine learning process, the use of at least one ensemble learning technique seemed to be a prudent choice. We selected the random forest model for it's ease of use and support of multi-output classes.
7. **Support Vector Machines:** The authors have a really solid theoretical understanding of how SVMs work so their inclusion was a natural choice. Because we have multi-class output, a support vector classifier using the "One-versus-One" multi-class strategy strategy was used.
8. **X-Gradient Boost:** With the guidance of our professor Anita Raja, we were directed to experiment with `XGBoost` as a possibly effective ensemble learning technique which might out perform the random forest. This is the model we had the least knowledge of, but it served as a great learning opportunity, both from a technical and theoretic perspective.

Model Selection

We performed model selection mainly by utilizing the `GridSearchCV` function. For each model we considered the smallest multi-class "tier list" of `[0, 5]` and performed a "wide and sparse search" over the parameter space. Based on the results of this initial model selection, we then conducted a more dense hyperparameter search, centered about the results from the initial sparse search. This two pass approach proved quite effective both in terms of runtime efficiency and classifier efficacy.

After model selection was complete for the multi-class label range `[0, 5]`, we applied the same hyperparameters to the larger "tier lists" with multi-class label ranges `[0, 10]` and `[0, 20]`. Unsurprisingly, since the models were not tuned for these larger multi-class labels ranges, their performance scores suffered significantly. We attempted to begin model selection for a the larger ranges, but this was not possible given the time constraints of the semester and our initial setbacks described below in the subsection **Challenges Faced**. The hyperparameters resulting from our model selection process can be found by referencing the executable file corresponding to the model listed in Table 15, and inspecting the definition of `hyperparameter_values` within the file.

Evaluation

We trained and cross validated each model with the hyperparameters obtained from model selection. Then we had each model train on the entire training set. Subsequently, each model was used to perform predictions on the testing set. The predictions were evaluated via the following metrics; *precision*, *recall*, and *F1 score*.

This training, prediction, and evaluation process was applied to the cross product of the model specification set and the tier list set. Phrased differently, for each combination of our 3 tier list multi-class label outputs and our 8 specified models, we evaluated our model's performance. The resulting measured scores of our evaluation metrics are listed in Table 17, Table 19, and Table 19.

Discussion

Experimental Results

Our three best performing models, when considering tier list `[1, 5]` were the K nearest neighbors, the multi-layer perceptron, and the X gradient boosting classifiers with F1 scores of `0.7375`, `0.7493`, and `0.7633`, respectively. On the larger tier lists `[0, 10]` and `[0, 20]`, all models performed poorly in roughly equal measure. Of all the models, the support vector classification appeared to degrade in performance the least and hence can be considered the most robust with respect to changing multi-class output ranges. The poor performance of the models when predicting for larger tier list is due to the lack of time to perform tailored model selection for them. Consequently, we will omit them from further discussion.

The predictive capabilities of the decision tree model were by far the least effective. This is not a surprising result, as we did not expect the model to be highly performant. To the contrary, a decision tree model with a strong predictive power would have been quite surprising and caused us to interpret the decision tree from insight into the structure of our dataset observations.

The Multinomial Naive Bayes model performed the second worst. This was a bit surprising at first. However after some consideration, we believe that the naive assumption tof each feature being independent of all others is likely false. Maybe features not dropped by the decorrelation pass still had "high" correlation within $[0.25, 0.6)$. This likely hampered the efficacy of the Naive Bayes model.

Logistic regression performed just shy of our top three models mentioned above. This suggests that there may be some linear correspondence between one or more of the features and the observed Elo labels. Further inspection of the model parameters may yield some insightful discoveries.

Similarly, the support vector machines assembled into a ove-vs-one classifier had very respectable, but not quite the best, performance. With further dedication to model selection, it may have easily reached the same caliber as our top three contenders.

The random forest model, along with the logistic regression and support vector machine models, can be considered in a "second place" ranking compared to the top performing K nearest neighbors, the multi-layer perceptron, and the X gradient boosting models. We beleive that more exploration with each model is warranted. However, we can sagfely drop the decision tree and naive Bayes models from consideration during the future work of performing model selection for "tier lists" `[0, 10]` and `[0, 20]`.

Overall, the predictive capabilities for small the smallest tier list is significant, though far from optimal. With roughly 75% "correctness" for a given metric, the top classifiers out perform any individual guessing in cases where the combat encounter's outcome is not a foregone conclusion. Further tuning of the models for the tier list `[0, 20]` would have allowed for a quantitative comparison of the disparity between the D&D source material's listed CR score $cr \in [1, 20] \subset \mathbb{N}$ and the classifier's predicted combat tier $pred \in [1, 20] \subset \mathbb{N}$. Future work, if pursued, will be directed towards this hypothesis testing, quantifiable and falsifiable observation.

Challenges Faced

Originally, the plan was to do a more in-depth analysis of magical items. This hit a snag when great difficulty was found with the automated combat simulator being used for the dataset generation. In particular, the libraries had many conflicts and missing sections. The uploader had accidentally pushed an experimental version, which called upon new functions not found in the original. Additionally, an entire section of the simulation would have remained broken, greatly reducing the accuracy of the predic-

tions in regards to creatures and classes with spell-casting capability. We scaled this back even further, attempting to simulate basic combat between brawlers and simple creatures, only to get heavily skewed results that were highly questionable, such as a greater than 70% loss rate for aboleths against itself out of 10000 runs. Similarly, team combat ended the same way, rendering even a simple simulation of basic mechanics completely useless.
Eventually, it was decided to use a dataset that already had computed values based on effectiveness. This use of "Elo rating system" is based on the same one used for competitive zero-sum games, and is relatively well understood. The source used their own `AI-based combat simulator` with working class features to generate the ELO ratings, based off of survival chances and victories scored by `millions of simulated fights` run by various individuals. Although the original goal had to be changed, it was done quickly with little down-time.

Lessons Learned

Most obviously, we learned that getting data in hand is important and that labeling a dataset can be a very arduous and time-prohibitive process. After pivoting from magic items to monsters, the time required to perform exploratory data analysis and data curation was significant, but not unexpected. Better time estimation for collection, cleaning, and labeling of data for ML techniques was a huge lesson.

Another large source of learning was from the model selection process. Initially, we had very little intuition as to which ML models would be effective for our dataset and our prediction objectives. The experimentation with eight very different modeling techniques provided us with a true breadth of experience from which we can draw from when undertaking further ML tasks.

Finally, the amount of clock-cycles required to perform thorough model selection. The hyperparameter searching process for just the first tier list of `[1, 5]` consumed all of our available time, given the short supply we had due to the magic item labeling set back. However, we are believe that the techniques we learned during our model selections, chiefly the “wide and sparse” search followed by a “targeted and dense” search, will be exceptionally valuable in our future endeavors, both in machine learning and as a transferable skill to other computationally expensive searching problems. With more available time, or simple choosing the monster rather than magic item dataset from the onset, we are confident that we could have tuned the models to perform well on the larger tier lists and performed hypothesis testing between the Challenge Rating system and our predictor(s). Both authors are interested in continued work on this project into the initial weeks of winter break to see if we can quickly get the project to a state that it is capable of hypothesis testing that the ML classifiers can out perform the published monster CR ratings.

In summary the authors gained multiple valuable insights into the application of and challenges with machine learning as a predictive problem solving technique. The production of functional classifiers and the rigorous evaluation via multiple metrics also demonstrates the authors' developed competency with machine learning techniques. We hope that the CSCI-795 Machine Learning seminar was at least one tenth as enriching for Professor Raja as it was for her students.

Bibliography

References

1. R. Rameshkumar and P. Bailey, "Storytelling with dialogue: A Critical Role Dungeons and Dragons Dataset," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 5121–5134. [Online]. Available: https://aclanthology.org/2020.acl-main.459

2. J. C. MacInnes, "The d&d sorting hat: Predicting dungeons and dragons characters from textual backstories," 2019.

3. P. Cavanaugh, "Machine learning for game master recommender," Ph.D. dissertation, University of Nebraska at Omaha, 2016.

4. F. G. Faria, L. T. Pereira, and C. F. M. Toledo, "Adaptive generation of characters for tabletop role playing games," in *2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. IEEE, 2019, pp. 39–46.

5. M. O. Riedl and V. Bulitko, "Interactive narrative: An intelligent systems approach," *Ai Magazine*, vol. 34, no. 1, pp. 67–67, 2013.

6. L. Ferreira and J. Whitehead, "Mtg: Context-based music composition for tabletop role-playing games," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 13, no. 1, 2017.

7. S. Risi and J. Togelius, "Increasing generality in machine learning through procedural content generation," *Nature Machine Intelligence*, vol. 2, no. 8, pp. 428–436, 2020.

8. R. R. Padovani, L. N. Ferreira, and L. H. Lelis, "Bardo: Emotion-based music recommendation for tabletop role-playing games," in *Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2017.

9. L. Ferreira, L. Lelis, and J. Whitehead, "Computer-generated music for tabletop role-playing games," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 16, no. 1, 2020, pp. 59–65.

10. L. J. Martin, S. Sood, and M. Riedl, "Dungeons and dqns: Toward reinforcement learning agents that play tabletop roleplaying games." in *INT/WICED@ AIIDE*, 2018.

Appendix

Table 6: **Initial Dataset**

#	Column Label	Data Type	#	Column Label	Data Type
0	Name	Textual	36	Immune Radiant	Binary
1	Type	Textual	37	Immune Slashing	Binary
2	Size	Integral	38	Immune Thunder	Binary
3	Armor	Integral	39	Resist Acid	Binary
4	Hit Points	Integral	40	Resist Bludgeoning	Binary
5	Move Burrow	Integral	41	Resist Cold	Binary
6	Move Climb	Integral	42	Resist Fire	Binary
7	Move Fly	Integral	43	Resist Force	Binary
8	Move Swim	Integral	44	Resist Lightning	Binary
9	Move Walk	Integral	45	Resist Necrotic	Binary
10	Stat Str	Integral	46	Resist Piercing	Binary
11	Stat Dex	Integral	47	Resist Poison	Binary
12	Stat Con	Integral	48	Resist Psychic	Binary
13	Stat Int	Integral	49	Resist Radiant	Binary
14	Stat Wis	Integral	50	Resist Slashing	Binary
15	Stat Cha	Integral	51	Resist Thunder	Binary
16	Save Str	Integral	52	Cause Blinded	Binary
17	Save Dex	Integral	53	Cause Charmed	Binary
18	Save Con	Integral	54	Cause Deafened	Binary
19	Save Int	Integral	55	Cause Frightened	Binary
20	Save Wis	Integral	56	Cause Grappled	Binary
21	Save Cha	Integral	57	Cause Incapacitated	Binary
22	Blind Sight	Binary	58	Cause Invisible	Binary
23	Dark Vision	Binary	59	Cause Paralyzed	Binary
24	Tremorsense	Binary	60	Cause Petrified	Binary
25	True Sight	Binary	61	Cause Poisoned	Binary
26	Immune Acid	Binary	62	Cause Prone	Binary
27	Immune Bludgeoning	Binary	63	Cause Restrained	Binary
28	Immune Cold	Binary	64	Cause Stunned	Binary
29	Immune Fire	Binary	65	Cause Unconscious	Binary
30	Immune Force	Binary	66	Multiattack	Binary
31	Immune Lightning	Binary	67	Spellcasting	Binary
32	Immune Necrotic	Binary	68	Damage Tags	Textual
33	Immune Piercing	Binary	69	Spellcasting Tags	Textual
34	Immune Poison	Binary	70	Trait Tags	Textual
35	Immune Psychic	Binary	71	Elo Rank	Integral

Table 9: **Fully Extracted Dataset**

#	Column Label	Data Type
0	Name	Textual
1	Type	Textual
2	Size	Integral
3	Armor	Integral
4	Hit Points	Integral
5	Move Burrow	Integral
6	Move Climb	Integral
7	Move Fly	Integral
8	Move Swim	Integral
9	Move Walk	Integral
10	Stat Str	Integral
11	Stat Dex	Integral
12	Stat Con	Integral
13	Stat Int	Integral
14	Stat Wis	Integral
15	Stat Cha	Integral
16	Save Str	Integral
17	Save Dex	Integral
18	Save Con	Integral
19	Save Int	Integral
20	Save Wis	Integral
21	Save Cha	Integral
22	Blind Sight	Binary
23	Dark Vision	Binary
24	Tremorsense	Binary
25	True Sight	Binary
26	Immune Acid	Binary
27	Immune Bludgeoning	Binary
28	Immune Cold	Binary
29	Immune Fire	Binary
30	Immune Force	Binary
31	Immune Lightning	Binary
32	Immune Necrotic	Binary
33	Immune Piercing	Binary
34	Immune Poison	Binary
35	Immune Psychic	Binary
36	Immune Radiant	Binary
37	Immune Slashing	Binary
38	Immune Thunder	Binary
39	Resist Acid	Binary
40	Resist Bludgeoning	Binary
41	Resist Cold	Binary
42	Resist Fire	Binary
43	Resist Force	Binary
44	Resist Lightning	Binary
45	Resist Necrotic	Binary
46	Resist Piercing	Binary
47	Resist Poison	Binary
48	Resist Psychic	Binary
49	Resist Radiant	Binary
50	Resist Slashing	Binary
51	Resist Thunder	Binary
52	Cause Blinded	Binary
53	Cause Charmed	Binary
54	Cause Deafened	Binary
55	Cause Frightened	Binary
56	Cause Grappled	Binary
57	Cause Incapacitated	Binary
58	Cause Invisible	Binary
59	Cause Paralyzed	Binary
60	Cause Petrified	Binary
61	Cause Poisoned	Binary
62	Cause Prone	Binary
63	Cause Restrained	Binary
64	Cause Stunned	Binary
65	Cause Unconscious	Binary
66	Multiattack	Binary
67	Spellcasting	Binary

#	Column Label	Data Type
68	Damage_A	Binary
69	Damage_B	Binary
70	Damage_C	Binary
71	Damage_F	Binary
72	Damage_I	Binary
73	Damage_L	Binary
74	Damage_N	Binary
75	Damage_O	Binary
76	Damage_P	Binary
77	Damage_R	Binary
78	Damage_S	Binary
79	Damage_T	Binary
80	Damage_Y	Binary
81	Spellcasting_CA	Binary
82	Spellcasting_CB	Binary
83	Spellcasting_CC	Binary
84	Spellcasting_CD	Binary
85	Spellcasting_CL	Binary
86	Spellcasting_CP	Binary
87	Spellcasting_CR	Binary
88	Spellcasting_CS	Binary
89	Spellcasting_CW	Binary
90	Spellcasting_F	Binary
91	Spellcasting_I	Binary
92	Spellcasting_P	Binary
93	Spellcasting_S	Binary
94	Aggressive	Binary
95	Ambusher	Binary
96	Amorphous	Binary
97	Amphibious	Binary
98	Antimagic Susceptibility	Binary
99	Brute	Binary
100	Charge	Binary
101	Damage Absorption	Binary
102	Death Burst	Binary
103	Devil's Sight	Binary
104	False Appearance	Binary
105	Fey Ancestry	Binary
106	Flyby	Binary
107	Hold Breath	Binary
108	Illumination	Binary
109	Immutable Form	Binary
110	Incorporeal Movement	Binary
111	Keen Senses	Binary
112	Legendary Resistances	Binary
113	Light Sensitivity	Binary
114	Magic Resistance	Binary
115	Magic Weapons	Binary
116	Pack Tactics	Binary
117	Pounce	Binary
118	Rampage	Binary
119	Reckless	Binary
120	Regeneration	Binary
121	Rejuvenation	Binary
122	Shapechanger	Binary
123	Siege Monster	Binary
124	Sneak Attack	Binary
125	Spell Immunity	Binary
126	Spider Climb	Binary
127	Sunlight Sensitivity	Binary
128	Turn Immunity	Binary
129	Turn Resistance	Binary
130	Undead Fortitude	Binary
131	Water Breathing	Binary
132	Web Sense	Binary
133	Web Walker	Binary
134	Elo Rank	Integral

Table 10: Dropped highly correlated features and their proxy

Proxy	Dropped Feature
Hit Points	Size
Hit Points	Stat Str
Hit Points	Stat Con
Stat Cha	Stat Int
Resist Fire	Resist Cold
Resist Fire	Resist Lightning
Resist Acid	Resist Thunder
Resist Acid	Incorporeal Movement
Damage Absorption	Immutable Form
Spell Immunity	Immune Force
Web Sense	Web Walker

Table 12: **Final Dataset**

#	Column Label	Data Type	#	Column Label	Data Type
0	Armor	Integral	48	Cause Paralyzed	Binary
1	Hit Points	Integral	49	Cause Petrified	Binary
2	Move Burrow	Integral	50	Cause Poisoned	Binary
3	Move Climb	Integral	51	Cause Prone	Binary
4	Move Fly	Integral	52	Cause Restrained	Binary
5	Move Swim	Integral	53	Cause Stunned	Binary
6	Move Walk	Integral	54	Cause Unconscious	Binary
7	Stat Dex	Integral	55	Multiattack	Binary
8	Stat Wis	Integral	56	Spellcasting	Binary
9	Stat Cha	Integral	57	Aggressive	Binary
10	Save Str	Integral	58	Ambusher	Binary
11	Save Dex	Integral	59	Amorphous	Binary
12	Save Con	Integral	60	Amphibious	Binary
13	Save Int	Integral	61	Antimagic Susceptibility	Binary
14	Save Wis	Integral	62	Brute	Binary
15	Save Cha	Integral	63	Charge	Binary
16	Blind Sight	Binary	64	Death Burst	Binary
17	Dark Vision	Binary	65	Devil's Sight	Binary
18	Tremorsense	Binary	66	False Appearance	Binary
19	True Sight	Binary	67	Fey Ancestry	Binary
20	Immune Acid	Binary	68	Flyby	Binary
21	Immune Bludgeoning	Binary	69	Hold Breath	Binary
22	Immune Cold	Binary	70	Illumination	Binary
23	Immune Fire	Binary	71	Immutable Form	Binary
24	Immune Lightning	Binary	72	Incorporeal Movement	Binary
25	Immune Necrotic	Binary	73	Keen Senses	Binary
26	Immune Piercing	Binary	74	Legendary Resistances	Binary
27	Immune Poison	Binary	75	Light Sensitivity	Binary
28	Immune Psychic	Binary	76	Magic Resistance	Binary
29	Immune Radiant	Binary	77	Magic Weapons	Binary
30	Immune Slashing	Binary	78	Pack Tactics	Binary
31	Immune Thunder	Binary	79	Pounce	Binary
32	Resist Bludgeoning	Binary	80	Rampage	Binary
33	Resist Cold	Binary	81	Reckless	Binary
34	Resist Force	Binary	82	Regeneration	Binary
35	Resist Necrotic	Binary	83	Rejuvenation	Binary
36	Resist Piercing	Binary	84	Shapechanger	Binary
37	Resist Poison	Binary	85	Siege Monster	Binary
38	Resist Psychic	Binary	86	Sneak Attack	Binary
39	Resist Radiant	Binary	87	Spell Immunity	Binary
40	Resist Slashing	Binary	88	Spider Climb	Binary
41	Cause Blinded	Binary	89	Sunlight Sensitivity	Binary
42	Cause Charmed	Binary	90	Turn Immunity	Binary
43	Cause Deafened	Binary	91	Turn Resistance	Binary
44	Cause Frightened	Binary	92	Undead Fortitude	Binary
45	Cause Grappled	Binary	93	Water Breathing	Binary
46	Cause Incapacitated	Binary	94	Web Walker	Binary
47	Cause Invisible	Binary	95	Elo Rank	Integral

Table 15: Model & file containing corresponding `hyperparameter_values` definition.

Model Name	Executable File
Decision Tree	`classification/model_DecisionTree.py`
K Nearest Neighbors	`classification/model_KNN.py`
Logistic Regression	`classification/model_LogisticRegression.py`
Multinomial Naive Bayes	`classification/model_NaiveBayes.py`
Multi-layer Perceptron	`classification/model_NeuralNetwork.py`
Random Forest	`classification/model_RandomForest.py`
Support Vector Machines	`classification/model_SVM.py`
X-Gradient Boost	`classification/model_XGBoost.py`

Table 17: Evaluation metrics for "Tier List" `[1, 5]`

	Precision	Recall	F1
Decision Tree	0.5974	0.5938	0.5941
K Nearest Neighbors	0.7433	0.7448	0.7375
Logistic Regression	0.7354	0.7344	0.7323
Multinomial Naïve Bayes	0.6501	0.6406	0.6425
Multi-layer Perceptron	0.7512	0.7500	0.7493
Random Forest	0.7305	0.7292	0.7222
Support Vector Classification	0.7370	0.7344	0.7326
X Gradient Boosting	0.7766	0.7656	0.7633

Table 19: Evaluation metrics "Tier List" `[1, 10]`

	Precision	Recall	F1
Decision Tree	0.3812	0.3839	0.3806
K Nearest Neighbors	0.4936	0.5071	0.4915
Logistic Regression	0.4972	0.5024	0.4924
Multinomial Naïve Bayes	0.4158	0.3791	0.3818
Multi-layer Perceptron	0.4427	0.4502	0.4380
Random Forest	0.4553	0.4502	0.4495
Support Vector Classification	0.4827	0.4882	0.4789
X Gradient Boosting	0.4642	0.4550	0.4565

Table 21: Evaluation metrics "Tier List" `[1, 5]`

	Precision	Recall	F1
Decision Tree	0.2214	0.2222	0.2159
K Nearest Neighbors	0.2702	0.2546	0.2513
Logistic Regression	0.2409	0.2315	0.2255
Multinomial Naïve Bayes	0.1302	0.1389	0.1217
Multi-layer Perceptron	0.2234	0.2269	0.2138
Random Forest	0.2796	0.2731	0.2719
Support Vector Classification	0.3092	0.3056	0.2999
X Gradient Boosting	0.2997	0.3056	0.2920