Predicting the Competitive Tier of Pokemon

Complex Random Forest and Feature Engineering Techniques, Giving New Tools to Gamers in the Age of Data Science

-Data Science growing in the video game industry

-Pokemon convenient for machine learning, lots of static data, turn-based, lots of untapped potential

Conclusion

The gaming industry has been growing exponentially. The number of active users tends to increase every minute and so does the overall income of the companies developing games. The inner infrastructure of the games gets even more complex providing more opportunities for players. An entirely new world and realities are created for the users. Top level visualization and design techniques, the latest visual effects, graphic elements and augmented reality effects provide customers with a high level of satisfaction.

Data science has entered various industries and improved the principles of their functioning forever. It has brought various businesses to a qualitatively new level of their development. The industry of gaming is no exception here. Moreover, data science techniques and methodologies have become integral parts of games development, design, operation, and many other stages of their functioning.

Abomasnow	Grass	Ice	Snow Warning	Soundproof	Untiered	90	92	75	92	85	60
Abra	Psychic		Inner Focus Synchronize	Magic Guard	LC	25	20	15	105	55	90
Absol	Dark		Justified Super Luck	Pressure	PU	65	130	60	75	60	75
Accelgor	Bug		Hydration Unburden	Sticky Hold	Untiered	80	70	40	100	60	145
Aegislash	Steel	Ghost	Stance Change		UU	60	50	140	50	140	60
Aegislash-Blade	Steel	Ghost	Stance Change		UU	60	140	50	140	50	60
Aerodactyl	Rock	Flying	Pressure Unnerve	Rock Head	NU	80	105	65	60	75	130
Aggron	Steel	Rock	Heavy Metal Sturdy	Rock Head	PU	70	110	180	60	60	50
Alakazam	Psychic		Inner Focus Synchronize	Magic Guard	UUBL	55	50	45	135	95	120
Alcremie	Fairy		Aroma Veil	Sweet Veil	Untiered	65	60	75	110	121	64
Alcremie-Gmax	Fairy		Aroma Veil	Sweet Veil	AG	65	60	75	110	121	64
Altaria	Dragon	Flying	Cloud Nine	Natural Cure	Untiered	75	70	90	70	105	80
Amaura	Rock	Ice	Refrigerate	Snow Warning	LC	77	59	50	67	63	46
Amoonguss	Grass	Poison	Effect Spore	Regenerator	UU	114	85	70	85	80	30
Anorith	Rock	Bug	Battle Armor	Swift Swim	LC	45	95	50	40	50	75
Appletun	Grass	Dragon	Gluttony Thick Fat	Ripen	Untiered	110	85	80	100	80	30
Appletun-Gmax	Grass	Dragon	Gluttony Thick Fat	Ripen	AG	110	85	80	100	80	30
Applin	Grass	Dragon	Bulletproof Ripen	Gluttony	LC	40	40	80	40	40	20
Araquanid	Water	Bug	Water Absorb	Water Bubble	NU	68	70	92	50	132	42
Arcanine	Fire		Flash Fire Justified	Intimidate	NU	90	110	80	100	80	95
Archen	Rock	Flying	Defeatist		LC	55	112	45	74	45	70
Archeops	Rock	Flying	Defeatist		PU	75	140	65	112	65	110
Arctovish	Water	Ice	Ice Body Water Absorb	Slush Rush	NUBL	90	90	100	80	90	55

Lots of fatalistic,

and not very useful,

analysis

Lessons Learned from Analyzing Competitive Pokemon

06 Sep 2018

Ultimately, this problem ended up being unsuitable for machine learning. Essentially, the feature space was too large and the interactions between the features were too complex for a model to learn given the small training set. In other words, there are many examples of Pokemon that are in their particular tier because of some unique combination of their type, ability, moves, and stats that are quite different from other Pokemon in that tier. You can read an in depth breakdown of this in the full notebook.

Identifying Legendary Pokémon using The Random Forest Algorithm

Sylvester Cardorelle May 31, 2019 · 7 min read

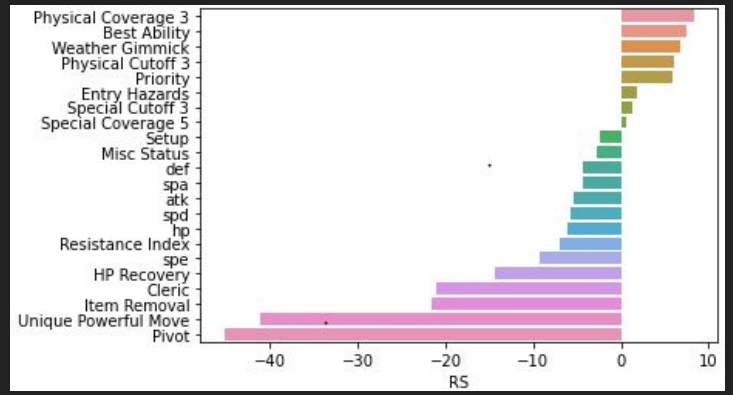
Unfortunately, there is no explicit criteria which defines these Pokémon.

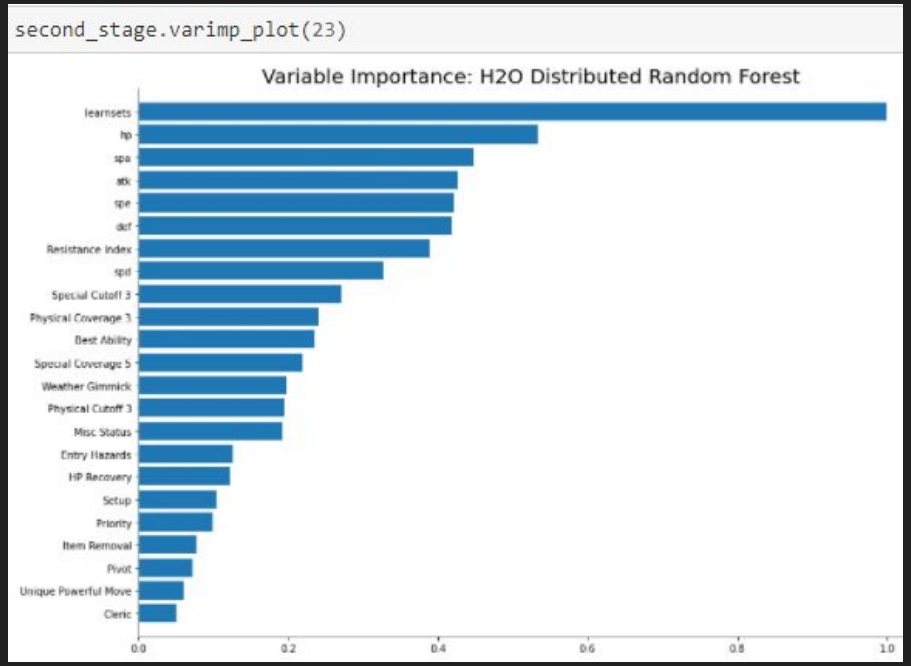
The only way to identify a Legendary Pokémon is through statements from official media, such as the game or anime.

Why might an accurate model of pokemon tiers be useful?

-game design, game quality, development time, design of generations

-feature importances: recommendations for players, powering products for gamers, design of single pokemon





Results!: Scores

H2O.ai 3.36.0.1

h2o.ai distributed random forest was best, with two stages

-training score: weighted balanced f-score is 0.887, f-scores of 7 classes range from 0.956 to 0.776

-test score: weighted balanced f-score is 0.776, f-scores of 7 classes range from 0.941 to 0.556 (and the lowest scoring classes tend to be quite small)

```
0.7761367825626139
```

Precision:
Recall:
F-Score:
Support:

```
(array([0.92105263, 0.45      , 1.        , 0.83333333, 1.        ,
        0.86666667, 0.94117647]),
 array([0.83333333, 0.84375   , 0.54545455, 0.41666667, 0.41666667,
        0.76470588, 0.94117647]),
 array([0.875     , 0.58695652, 0.70588235, 0.55555556, 0.58823529,
        0.8125    , 0.94117647]),
 array([84, 32, 11, 12, 12, 17, 17], dtype=int64))
```

Benchmark Scores:

```
0.48060193774479487
```

Okay, so our classifier isn't doing great - it averages around 50% accuracy. Let's take a look into what sort of cases it's missing.

```
tiers_comp = ["LC+NFE", "Untiered+PU+NU+RU", "UU+OU+Uber+AG"]
```

```
0.7746753246753246
```

As expected, our accuracy drastically increases when we reduce the cardinality of tiers, reaching an average of around 70%. This is unfortunately still quite low. Furthermore, we haven't really improved anything, we're simply making the problem easier for the classifier.

More Benchmark Scores:

Competitive Pokemon Usage Tier Classification

Devin Navas
Fordham University
dnavas1@fordham.edu

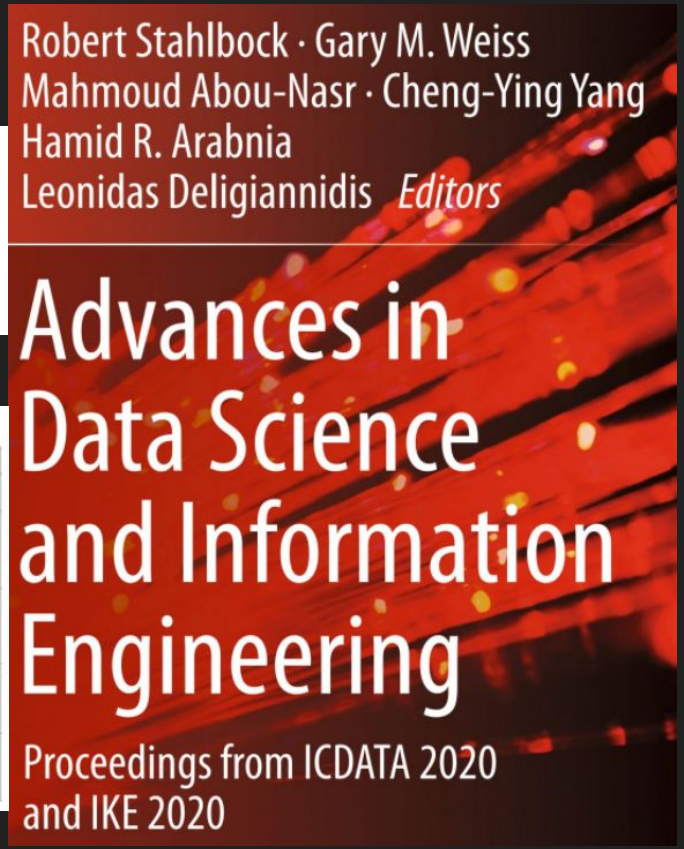
Dylan Donohue
Fordham University
ddonohue7@fordham.edu

TABLE IV. PRECISION OF CLASSIFICATIONS- TEST SET WITHOUT UNDERSAMPLING

Usage Tier	Algorithm		
	J48	*Lazy IBk*	*Logistic Regression*
Ubers	.909	1.00	.900
OU	.731	1.00	.478
UUBL	.500	1.00	0.00
UU	.763	1.00	.387
RUBL	0.00	1.00	0.00
RU	.941	1.00	.438
NUBL	.500	1.00	0.00
NU	.840	1.00	.375
PU	.796	1.00	.475

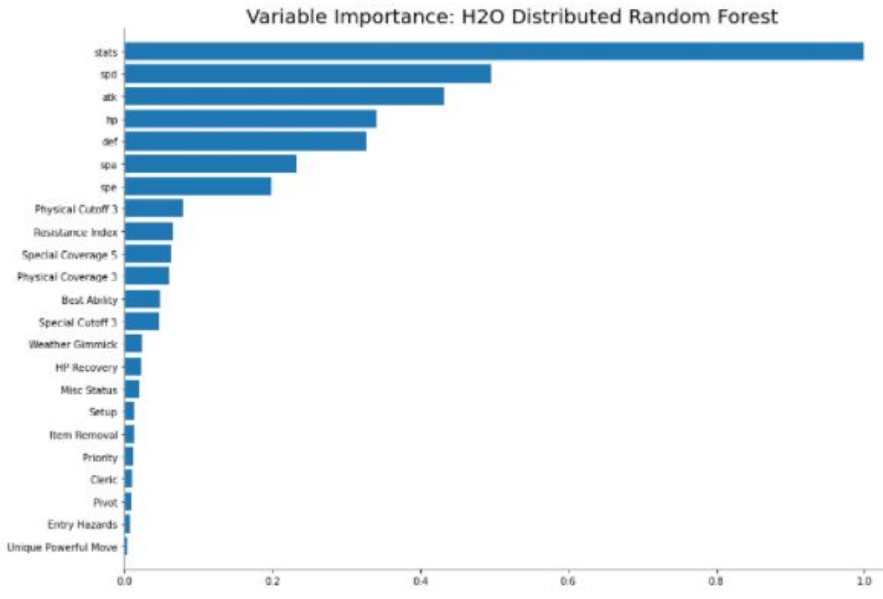
TABLE I. ACCURACY OF CLASSIFICATIONS(%)

Methods	Algorithm		
	J48	*Lazy IBk*	*Logistic Regression*
Cross Validation without Undersampling	32.32	29.60	32.65
Cross Validation with Undersampling	26.00	27.00	29.50
Test Set without Undersampling	78.85	100.0	47.40
Test Set with Undersampling	64.00	78.28	47.42

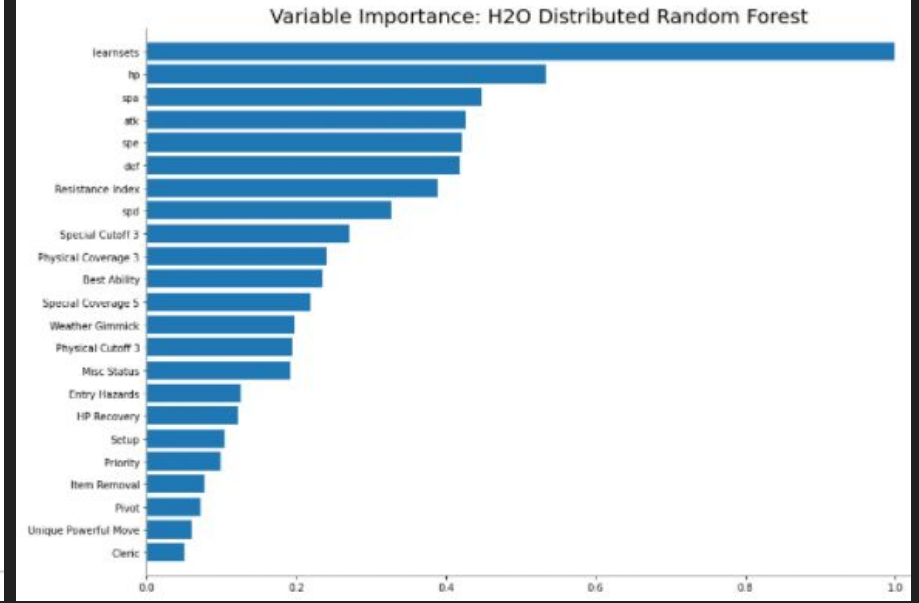


More Results!: Feature Importances

```
first_stage.varimp_plot(23)
```

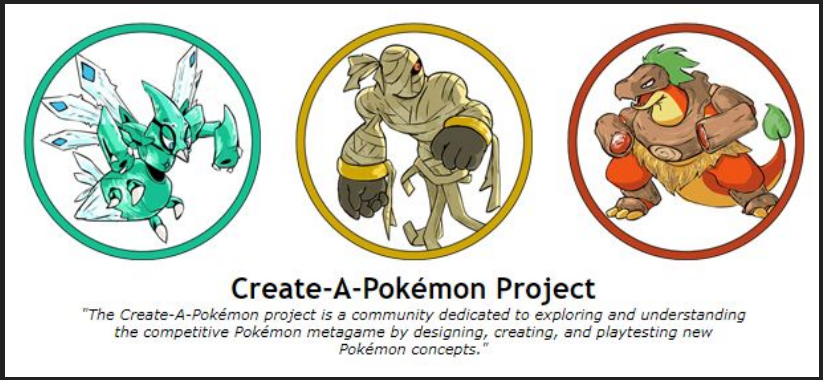



```
second_stage.varimp_plot(23)
```



What To Do With This Model?

Exactly what I mentioned earlier:



-game design, game quality, development time, design of generations

-feature importances: recommendations for players, powering products for gamers, design of single pokemon

Approach

The 6v6 metagame

Mekkah · Oct 12, 2010

```
pokemon_df['formats'].unique()
```

```
array(['LC', 'NFE', 'RUBL', 'PU', 'NU', 'Untiered', 'UU', 'OU', 'UUBL',
       'PUBL', 'RU', 'Uber', 'NUBL'], dtype=object)
```

```
ZU      335
PU      153
OU       57
Uber     56
RU       51
UU       45
NU       41
Name: Change, dtype: int64
```

Smogon-wide Clauses

English Español Français Italiano Portug

There are several clauses that apply to most offi

Species Clause

A player cannot have two Pokemon with th

- For example, using a team with two Koffing wo
- Using different formes on the same team, such Pokédex number (#479).

Sleep Clause

If a player has already put a Pokemon on t to sleep.

Evasion Clause

A Pokemon may not have either Double Te

OHKO Clause

A Pokemon may not have the moves Fissu

Moody Clause

A team cannot have a Pokemon with the a

Endless Battle Clause

Players cannot intentionally prevent an opp

Web Scraping

SCRAPING DATA FROM A JAVASCRIPT WEBPAGE WITH PYTHON

```
<!doctype html>
<html>
  <head>
    <title></title>
    <base href="/dex/">
    <link rel="stylesheet" href="media/build/dex.css.v.cuEUhCNveq-
    <script src="https://hb.vntsm.com/v3/live/ad-manager.min.js" t
    <meta name="viewport" content="width=device-width, initial-sca
    <meta name="fragment" content="!">

        <script type="text/javascript">
            dexSettings = {"injectRpcs":[["[\"dex\",\"dump-gens\"]
        </script>
    </head>
  <body tabindex="-1" class="theme--light">
    <div id="rich-media-placement"></div>
    <div id="container"></div>
    <div id="loading">
        <div class="spinner">
            <div class="spinner-img"><img src="media/images/loadin
            <div class="spinner-text">Loading...</div>
        </div>
    </div>
```

Data Cleaning

https://regex101.com

regex101: build, test, and debug regex

Regular expression tester with syntax highlighting, explanation, cheat sheet for PHP/PCRE, Python, GO, JavaScript, Java. Features a regex quiz & library.

http://jsonviewer.stack.hu

Online JSON Viewer

JSON Viewer - Convert JSON Strings to a Friendly Readable Format.

```
strategies_dict = {}

strategydexcopy = strategydex_df.copy()

for pokemon in pokemon_df.index:
    pokemon_access = pokemon.replace("'", "").replace
    dex_entry = strategydexcopy.loc[strategydexcopy['
    format_list = []
    for competitive_format in dex_entry['strategies']
        format_dict = {}
        current_format = competitive_format['format']
        if current_format in acceptable_formats:
            moveset_list = []
            for moveset in competitive_format['movese
                moveset_dict = {}
                moveset_name = moveset['name']
                move_list = []
                for move in moveset['moveslots']:
                    for entry in move:
                        move_list.append(entry['move'
                moveset_dict[moveset_name] = move_lis
                moveset_list.append(moveset_dict)
            format_dict[current_format] = moveset_lis
            format_list.append(format_dict)
    strategies_dict[pokemon] = format_list
```

```
ed_data/PokemonData2021.csv", encoding="utf8") as infile:


.str.replace('<p>.+?<\/p>', '', regex=True)
.str.replace('<section>.+?<\/section>', '', regex=True)
.str.replace('<h1>.+?<\/h1>', '', regex=True)
.str.replace('\\\\n', '', regex=True)
.str.replace("\'", '\"')
.str.replace('King\"s', "King\'s")
.str.replace('Maki\"s', "Maki\'s")
.str.replace('Land\"s', "Land\'s")
.str.replace('Dragon\"s', "Dragon\'s")
.str.replace('Sirfetch\"d', "Sirfetch\'d")
.str.replace('Farfetch\"d', "Farfetch\'d")
.str.replace('Nature\"s', "Nature\'s")
.str.replace('Forest\"s', "Forest\'s")
.str.replace('drampa\"s', "drampa\'s")
.str.replace('False', '\"\"')
.str.replace('True', '\"\"')
.str.replace('None', '\"\"')
.str.replace('\"\"\"\s', '\"', regex=True)
.str.replace('<ul>.+?<\/ul', '', regex=True)
.str.replace('Swipe', 'False Swipe') #newline
.str.replace('Surrender', 'False Surrender') #newline
.apply(json.loads)
```

Big Problem!!!

738 rows × 14 columns

738 rows × 18 columns

738 rows × 242 columns

738 rows × 644 columns

14 + 18 + 242 + 644 = 918!!!

degrees of freedom = number of independent values – number of statistics

… models cannot be used "out of the box", since the standard fitting algorithms all require p<n; in fact the usual rule of thumb is that there be five or ten times as many samples as variables.

Resistance Index = Resists + 2(Strong Resists) + 4(Immune) - Weaknesses - 2(Strong Weaknesses)

×	Defending type																	
Attacking type	NORMAL	FIGHT	FLYING	POISON	GROUND	ROCK	BUG	GHOST	STEEL	FIRE	WATER	GRASS	ELECTR	PSYCHC	ICE	DRAGON	DARK	FAIRY
NORMAL	1×	1×	1×	1×	1×	½×	1×	0×	½×	1×	1×	1×	1×	1×	1×	1×	1×	1×
FIGHT	2×	1×	½×	½×	1×	2×	½×	0×	2×	1×	1×	1×	1×	½×	2×	1×	2×	½×
FLYING	1×	2×	1×	1×	1×	½×	2×	1×	½×	1×	1×	2×	½×	1×	1×	1×	1×	1×
POISON	1×	1×	1×	½×	½×	½×	1×	½×	0×	1×	1×	2×	1×	1×	1×	1×	1×	2×
GROUND	1×	1×	0×	2×	1×	2×	½×	1×	2×	2×	1×	½×	2×	1×	1×	1×	1×	1×
ROCK	1×	½×	2×	1×	½×	1×	2×	1×	½×	2×	1×	1×	1×	1×	2×	1×	1×	1×
BUG	1×	½×	½×	½×	1×	1×	1×	½×	½×	½×	1×	2×	1×	2×	1×	1×	2×	½×
GHOST	0×	1×	1×	1×	1×	1×	1×	2×	1×	1×	1×	1×	1×	2×	1×	1×	½×	1×
STEEL	1×	1×	1×	1×	1×	2×	1×	1×	½×	½×	½×	1×	½×	1×	2×	1×	1×	2×
FIRE	1×	1×	1×	1×	1×	½×	2×	1×	2×	½×	½×	2×	1×	1×	2×	½×	1×	1×
WATER	1×	1×	1×	1×	2×	2×	1×	1×	1×	2×	½×	½×	1×	1×	1×	½×	1×	1×
GRASS	1×	1×	½×	½×	2×	2×	½×	1×	½×	½×	2×	½×	1×	1×	1×	½×	1×	1×
ELECTR	1×	1×	2×	1×	0×	1×	1×	1×	1×	1×	2×	½×	½×	1×	1×	½×	1×	1×
PSYCHC	1×	2×	1×	2×	1×	1×	1×	1×	½×	1×	1×	1×	1×	½×	1×	1×	0×	1×
ICE	1×	1×	2×	1×	2×	1×	1×	1×	½×	½×	½×	2×	1×	1×	½×	2×	1×	1×
DRAGON	1×	1×	1×	1×	1×	1×	1×	1×	½×	1×	1×	1×	1×	1×	1×	2×	1×	0×
DARK	1×	½×	1×	1×	1×	1×	1×	2×	1×	1×	1×	1×	1×	2×	1×	1×	½×	½×
FAIRY	1×	2×	1×	½×	1×	1×	1×	1×	½×	½×	1×	1×	1×	1×	1×	2×	2×	1×

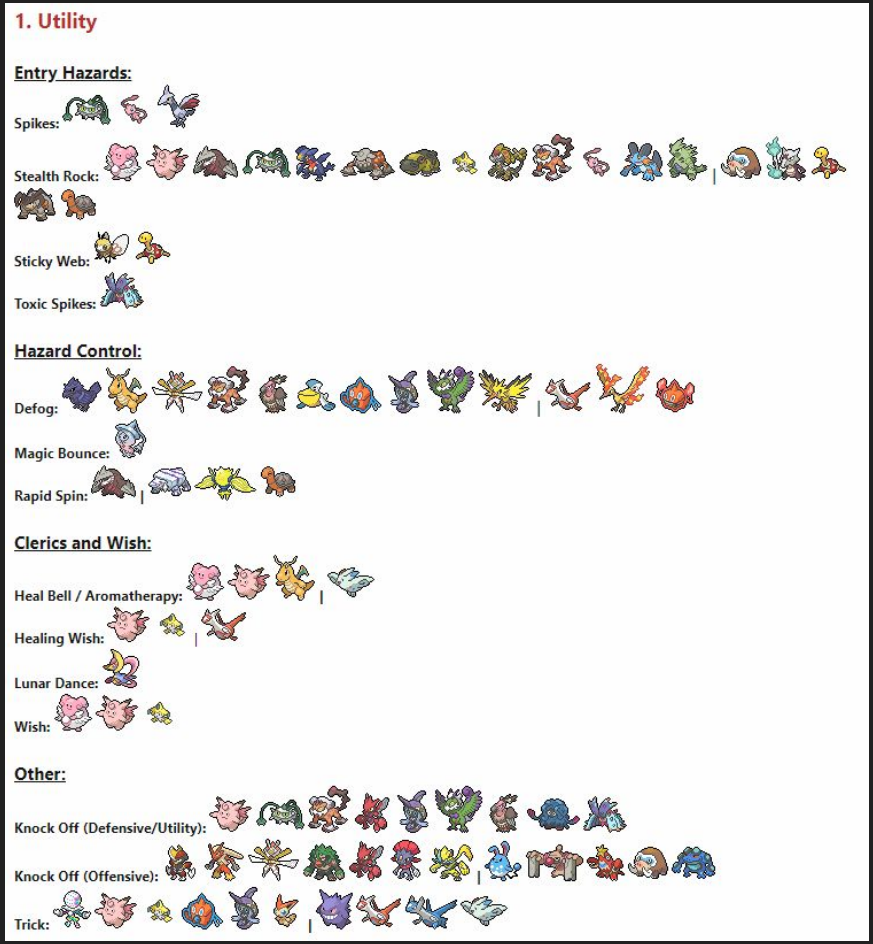
These matchups are suitable for Generation VI onward.

Roles in Competitive Pokemon

-single out important moves and abilities

-allow for surprising niches, in spite of things like weak stats, etc.

-model could miss this easily without features

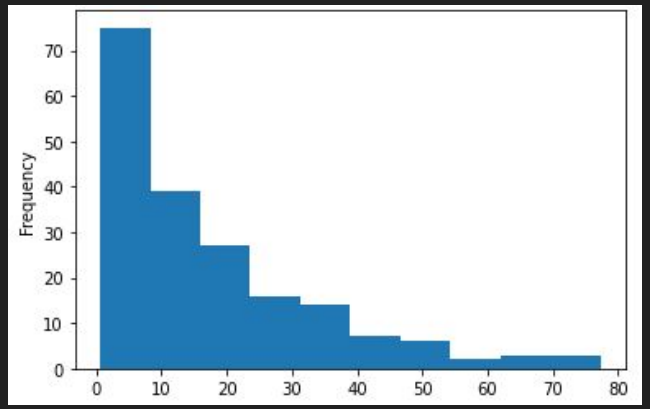


“C-index”: Competitiveness Index for Moves and Abilities

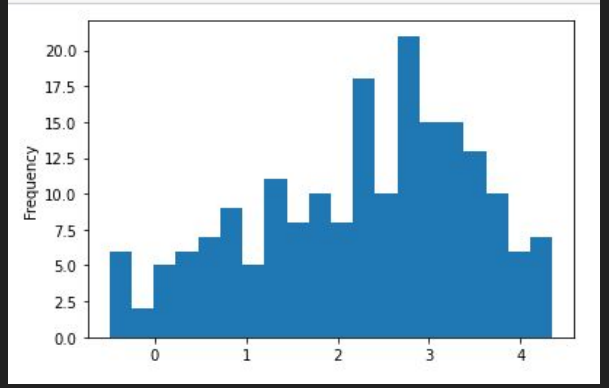
738 rows × 242 columns

738 rows × 644 columns

TF-IDF (**term frequency-inverse document frequency**) is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents. May 10, 2019




```
cindex_distro_log = cindex_distro.apply(np.log)
cindex_distro_log.plot.hist(by='cindex', bins=20)
plt.show()
```



More About the Cindex

```
In [95]: n_learnset[np.abs(stats.zscore(n_learnset)) > 3]

Out[95]: 1.0    77
         2.0    28
         Name: n_learnset, dtype: int64
```

```
In [96]: n_learnset[np.abs(stats.zscore(n_learnset)) > 2]

Out[96]: 1.0    77
         2.0    28
         Name: n_learnset, dtype: int64
```

Unique Powerful Moves Feature

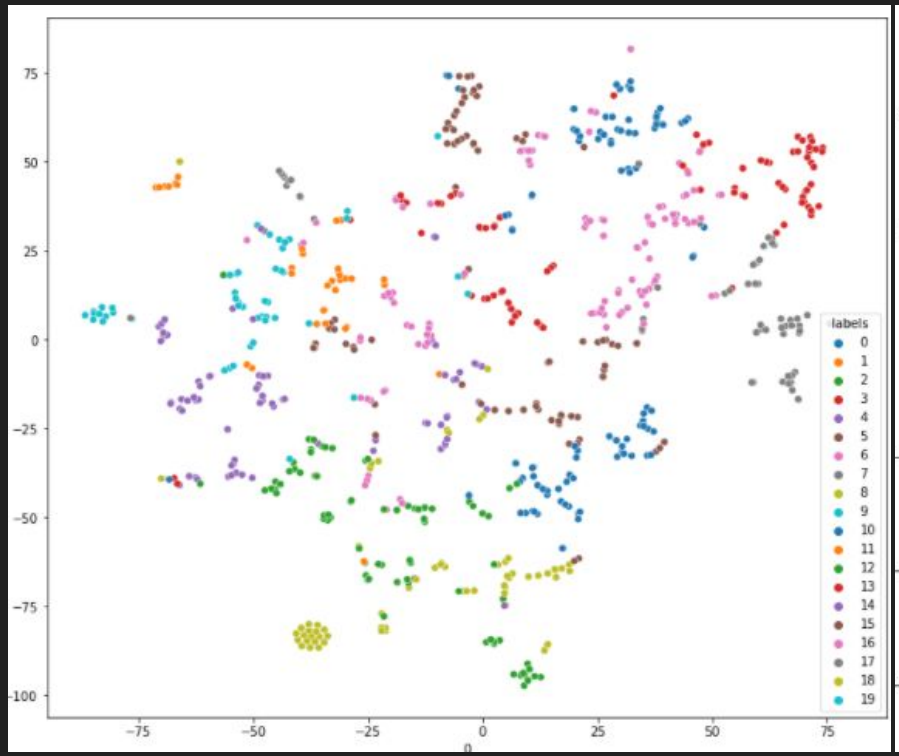
Now we need to make a feature indicating wh
moves with cindex 100 and n_learnset 1 or 2

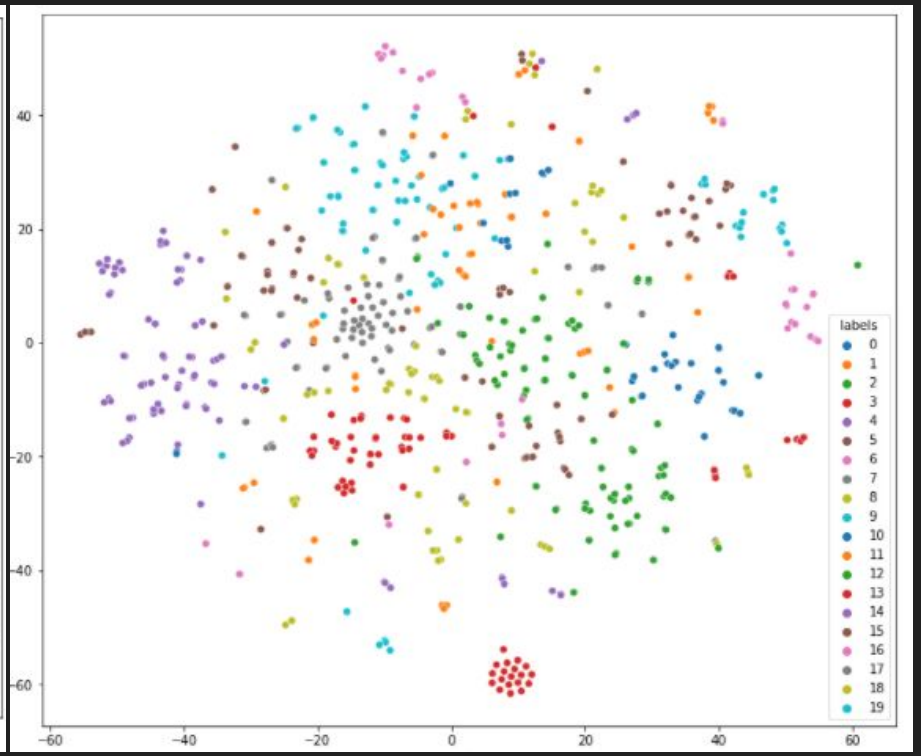
Aggregate Feature: Physical Cutoff
Aggregate Feature: Physical Coverage
Aggregate Feature: Special Cutoff
Aggregate Feature: Special Coverage
Aggregate Feature: Miscellaneous Status

```
sical Cutoff 5', 'Physical Cutoff 6']].corrwith(pokemon_data['format codes'])
```

```
Physical Cutoff 1    0.270197
Physical Cutoff 2    0.284223
Physical Cutoff 3    0.324136
Physical Cutoff 4    0.325190
Physical Cutoff 5    0.300717
Physical Cutoff 6    0.276238
dtype: float64
```

Clusters: Shrinking the Decision Space





Models We Tried

-Logistic Regression, Decision Tree: Far lower accuracy on the train and test sets

-KNN: performed almost as well as Random Forest (similar to clustering)

-KNN only gives predictions, not feature importances or coefficients

Are categorical variables getting lost in your random forests?

less than 1 minute read

I've been one-hot encoding categorical variables for as long as I have been using sci-kit learn. It turns out that you can lose a lot of predictive power this way, and that alternatives do exist.

> *Decision tree models can handle categorical variables without one-hot encoding them. However, popular implementations of decision trees (and random forests) differ as to whether they honor this fact. We show that one-hot encoding can seriously degrade tree-model performance. Our primary comparison is between H2O (which honors categorical variables) and scikit-learn (which requires them to be one-hot encoded).*

H2O.ai

3.36.0.1

Docs » Algorithms » Distributed Random Forest (DRF) Edit

Distributed Random Forest (DRF)

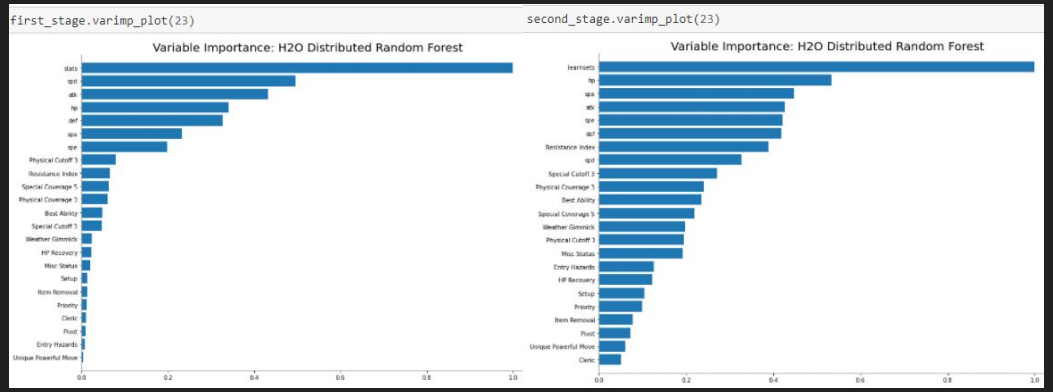
Our most important feature (proven by feature importances) was clustering, which is categorical, so it may have been essential that h2o could properly assign importance to it!

Improved ability to train on categorical variables (using the `nbins_cats` parameter)

Two-Stage Modeling!

significantly improved accuracy, and especially helped with overfitting

```
ZU      335
PU      153
OU       57
Uber     56
RU       51
UU       45
NU       41
Name: Change, dtype: int64
```



-model for the higher 6 classes has more balanced feature importances

-also tried three-stage, due to size of PU, but this did not improve accuracy on training or test data

Future Ideas!