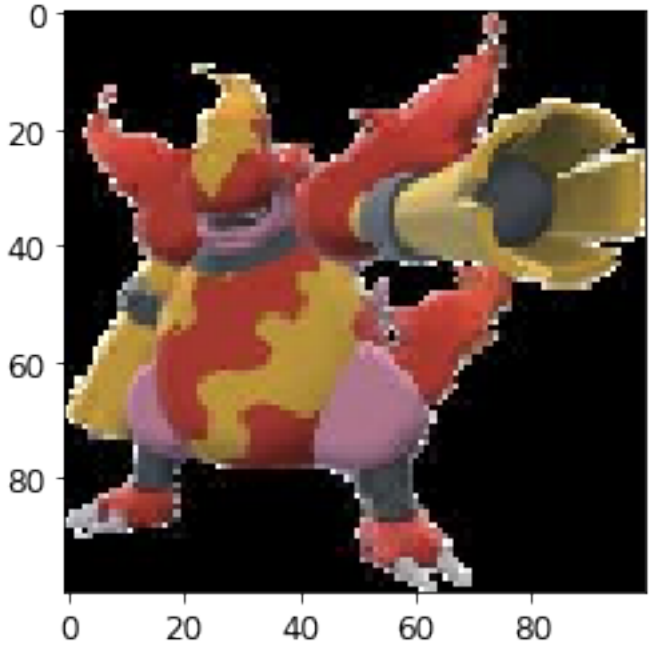
Kai Mcconnell
Project 2 Regression Analysis
Pokemon Data Set

I looked through some of the datasets offered on Kaggle and came across a dataset on Pokemon images and their strengths. I thought this seemed like an interesting data set as it involved images like the 3rd example notebook we explored in class.

My first steps were to examine the data set and see what features it had available however I quickly realized this data set had limited features. The only features I had to work with were the images, the names of the Pokemon, and their strength value. This made me unsure whether this would be a good data set to use random tree classifiers on however, I had already put some work in so I decided to explore what I could get out of this dataset. The first steps I undertook was to see what my features looked like. What I found was images like this:

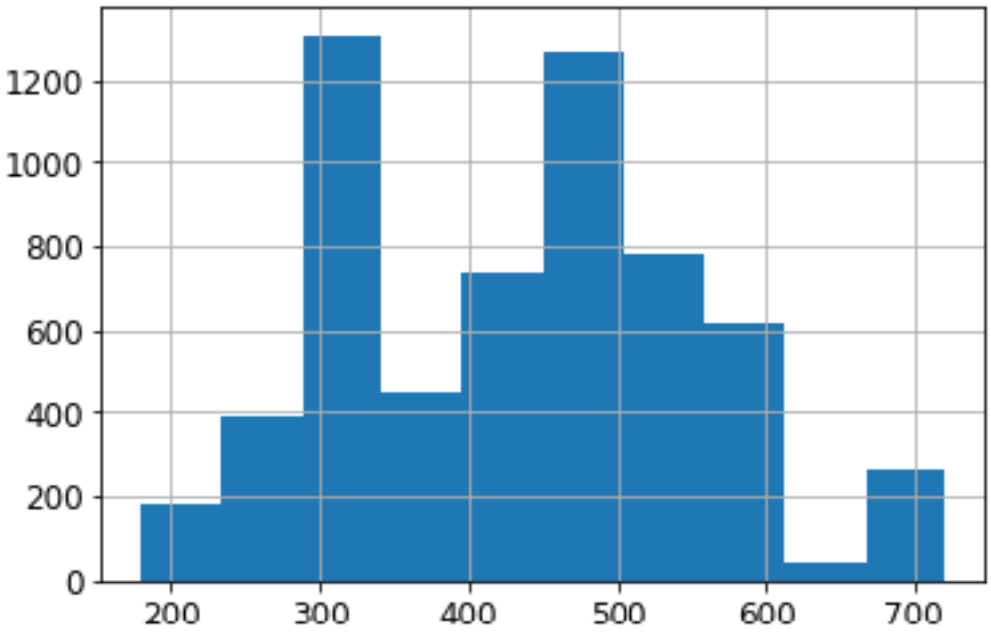


I also found a csv file with this data:

	name	strength
30	Absol	465
31	Absol	465
32	Absol	465
33	Absol	465
34	Accelgor	495
35	Accelgor	495
36	Aegislash	520
37	Aegislash	520
38	Aegislash	520
39	Aegislash	520

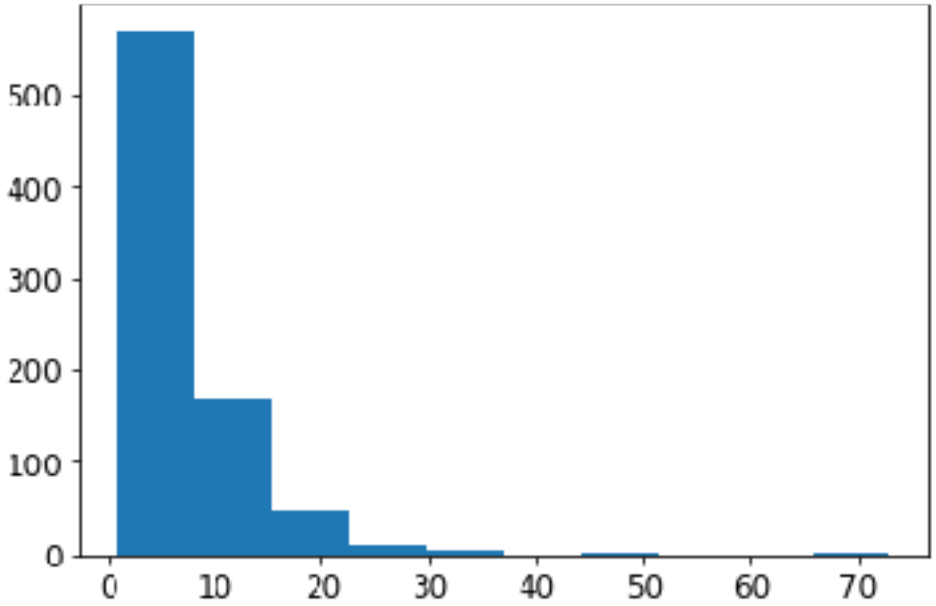
After this examination I started to think about what I wanted to solve as part of my machine learning project. I found a couple of options. I could try to predict the name of the Pokemon using the images, or the strength values or both. Or I could do the reverse and use the image of the Pokemon to predict its strength. Since the person who put the data set up on Kaggle had already explored the predictive nature of Pokemon images on their individual strengths I decided to go for the Pokemon name prediction idea. In analyzing the two features I had to predict the name of the Pokemon I wasn't sure how best to analyze the image so I focused on analyzing the strength feature. What I found wasn't particularly promising as the distribution of strength across Pokemon was over a fairly small range and therefore would not be incredibly effective as a predictor. This is shown in this histogram:

Strength Distribution



Because there are noticeable locations where a large amount of the data has a similar strength it is unlikely the model will be able to successfully predict the Pokemon frequently. Since strength was off the table I decided to use the image and see what results I could get. The image data had a great deal of complexity and depth to it which made it difficult to analyze so I did my best to explore the data around it. A thought I had was that the more data of each Pokemon I had the more accurate the model would be at predicting it so I also looked at the distribution of Pokemon. My exploration of the distribution cumulated into this histogram:

Distribution of Pokemon



Upon seeing this histogram I realized that most of the Pokemon had under 15 data points. I saw a few outliers and decided to see which Pokemon had the most references. These Pokemon were:

Pikachu: 73
Eevee: 46
Minior: 36

Which indicates they would be the most likely to be successfully predicted by the model.

After completing this analysis of my data points I started training and testing. I started out with a 80% train to 20% test. This resulted in these values:

Number of train to test
4829
1207

Each piece of data looked something like this before splitting into training and testing

Ninjask
[0. 150. 141. ... 0. 0. 456.]

The name was the goal and the array of pixels were the features used to achieve predicting the goal. If you notice at the end of the array is a larger number which looks out of place. This value is the strength of the Pokemon. After initially throwing out the strength feature I realized I shouldn't throw out data that might help improve the accuracy of my model. These were the results:

Just Image:
Accuracy: 0.1805960264900662
With Strength:
Accuracy: 0.188069594034797

At first ignored strength as it appeared as though it wouldn't be a useful feature due to large quantities of the data having the same power level. After analyzing the data using just image processing decided to see if strength affected the accuracy at all and had my hunch proven correct as the data provided little meaningful improvement to the accuracy of the model. I then went back and looked at ways I could improve my model. I looked into hyper-parameter optimization however because I had such limited data I found it wasn't particularly useful or relevant. If 1 of the features was the goal then there were only 2 features you could use to try to predict the goal. I did find some success in using cross validation on my dataset. This revealed to me that I had been slightly under fitting. The result of my doing so however was fairly minimal as it increased the accuracy of my model by 3% or so.

Cross validation on Data set
Accuracy: 0.21274834437086093

Overall this data set as I suspected was not the best choice for a random forest classifier problem however I learned some weaknesses of the random forest classifier as I explored this data set and will know some signs to look out for in the future to avoid it. In the future I think I would use an auto encoder for this type of problem. From the research I did it seemed much more useful for generalizing images enough to be useful for predicting values.