MATH 637 - Mathematical Techniques for Data Science Final Project: Visualization and Prediction of Pokemon and Corresponding Battles

Alexandra Vásconez-Moncayo

June 2021

1 Introduction

I set out on this project to see if there was a way to develop some form of algorithm to determine the best Pokemon to choose to battle with. My goal was to be able to use this same algorithm for all existing Pokemon, along with any that may be created in the future. There were some notable differences between the Pokemon that tended to lose most battles compared to those who won the majority of theirs, but the pattern seemed to not be as discernible for those Pokemon in the middle of the pack. At the end though, my training and test data sets earned high accuracy scores using different techniques to classify and predict.

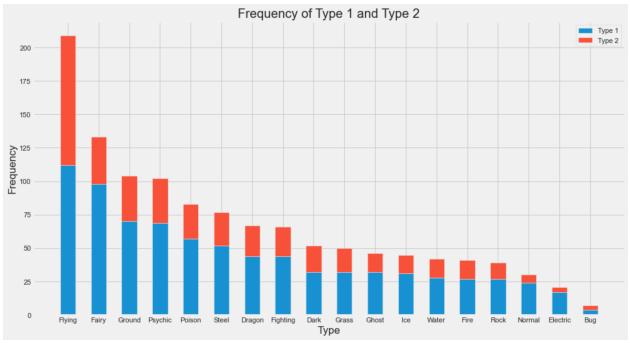
Here, I will detail the process of visualizing the different classifications of Pokemon, including by type, generation, and other important statistics they have. We first explore the data sets imported which includes Pokemon names and details, followed by combat information for the corresponding Pokemon. We use this opportunity to ensure that errors are minimized by assuring that all relevant information is included. Later, we use this to visualize how unique the Pokemon are, by using different methods of plotting the data.

Finally, we split the data into relevant training and testing sets in order to check how accurate different methods of prediction classification processes are.

2 Data Exploration

We begin with importing the aforementioned data sets, `pokemon.csv` and `combat.csv` while taking note that there should be 800 different types of Pokemon with about 10 classifications. We also note that only about half of them have two Types associated with them.

```
Name          799
Type 1        800
Type 2        414
HP            800
Attack        800
Defense       800
Sp. Atk       800
Sp. Def       800
Speed         800
Generation    800
Legendary     800
dtype: int64
```



3 Data Preprocessing

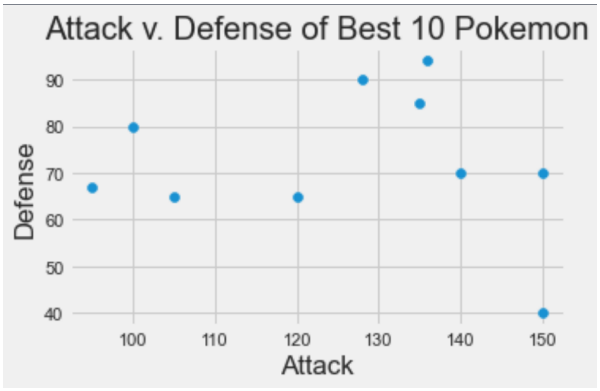
It is here that we notice one Pokemon that seems to be missing a name. Therefore, using the relevant code to find the missing name and then an online Pokedex in order to match the Pokemon to its corresponding number: Primeape. We also work to include more relevant details needed to better visualize and eventually form predictions on which Pokemon have the best chance of winning combats. So, we come up with a "Winning Percentage" column and include this in our original dataset. With this, we come across new information: there appears to be a Pokemon that has a 0% winning percentage. With some more exploring, we find out the losing creature is Shuckle: this is a good time to note that one should probably never pick them to battle with.

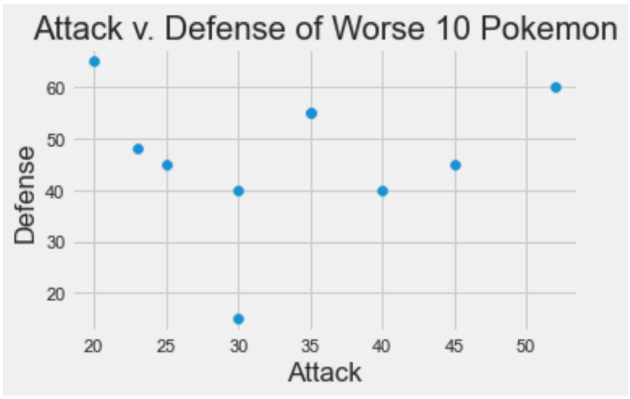
```
Looking at the dimensions of our dataframes
Count by first winner shape: (784, 2)
Count by second winner shape: (784, 2)
Total Wins shape : (783,)
```

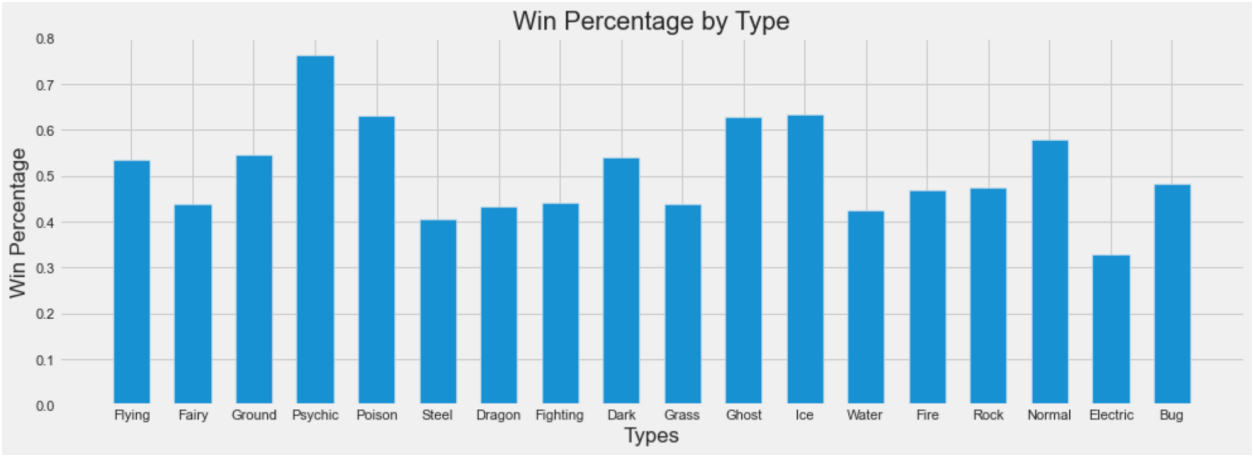
Name	Shuckle
Type 1	Bug
Type 2	Rock
HP	20
Attack	10
Defense	230
Sp. Atk	10
Sp. Def	230
Speed	5
Generation	2
Legendary	False

4 Data Visualization

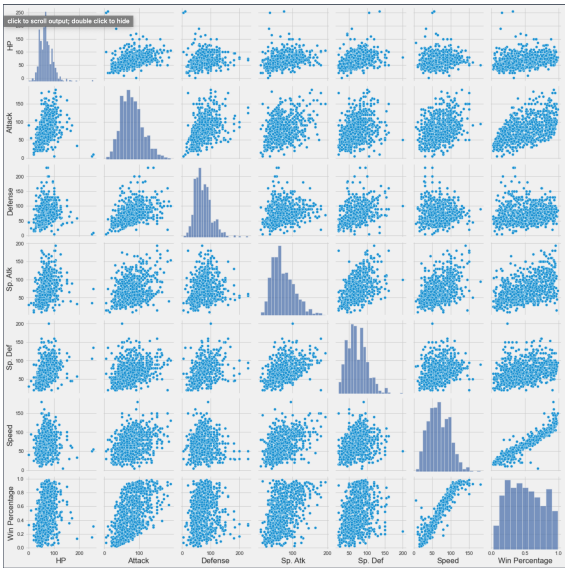
This is now the time to look at the new information we found and visualize our data with the newly added column.

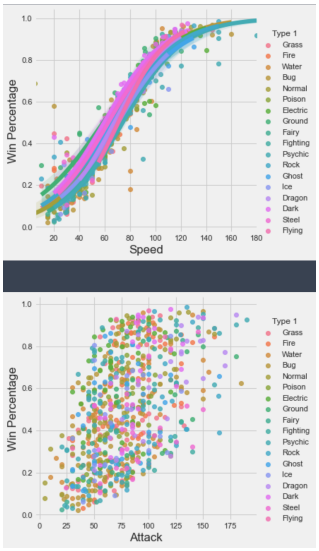






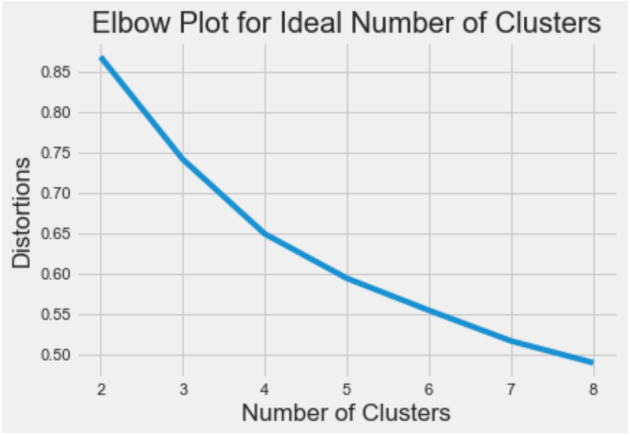
It is here that I also attempted to find a connection between the different classifications:





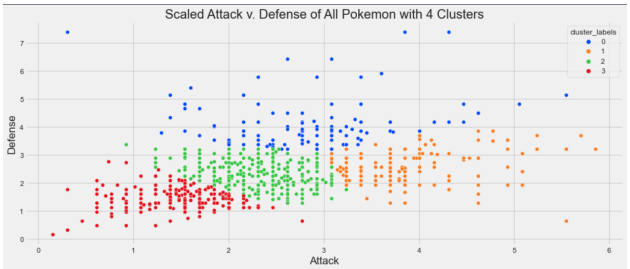
5 Algorithm Applications

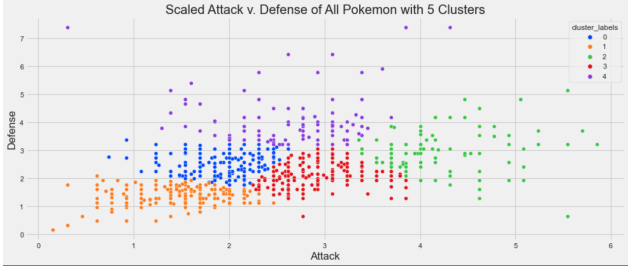
Here is when we prepare the data in order to apply our algorithms. We begin by attempting to cluster the data to see if there's a noticeable pattern with scaled data of our "Attack" and "Defense" columns.



Note there is not much as much of an elbow as we would have hoped for. Nonetheless, I used 4 and 5 clusters to see the best fit for myself. We did use a scaled version of our data set, to try and make this as accurate as possible.

It is here where I abandoned the clustering method since it did not seem to be of much use.



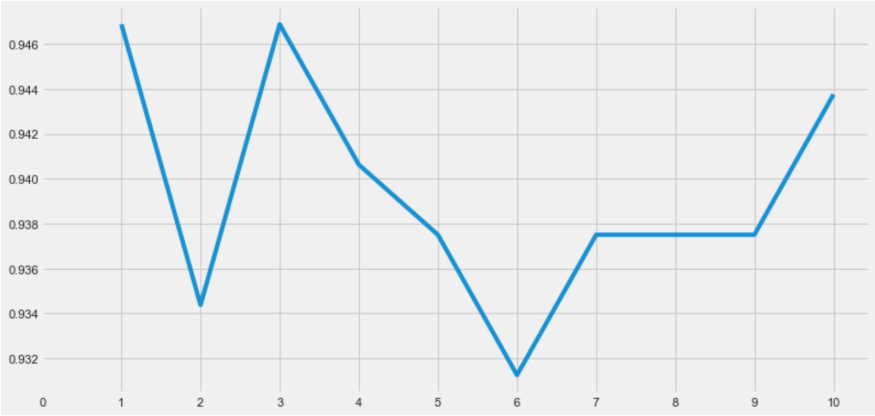


Next, I attempted to split the data using the *sklearn* model selection into corresponding training and testing set with a random state of 39. We compare the following models based on their accuracy

- Linear SVM
- KNN
- Random Forest Classifier
- Logistic Regression

For Linear SVM, we test this model with four values of $C = 0.001, 0.01, 0.1, 1$. This gives us a maximum accuracy of 96.56%.

For KNN Classification, we chose this as it is one of the simplest models for supervised learning. Therefore, we spend less time on training with a trade-off for speed in testing. We tested this with ranging values from 1 to 10, inclusive. This leads to an accuracy of 94.69%.



I chose to use Random Forest Classification and purposefully excluded the Decision Tree Classifier since the former would provide a more rigorous and stable prediction, since

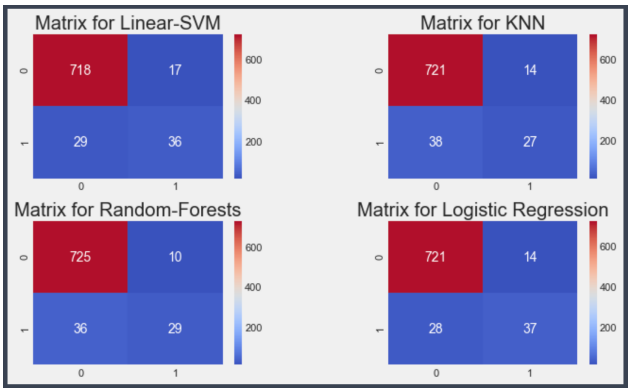
it merges many uncorrelated decision trees together. Since this model adds additional randomness, we obtain more diverse results which lead to better accuracy, a max value 95.94&.

Finally, I wanted a model that would perform well even if the data used to compare features were outside the range of the training data. Hence, Logistic Regression came into play, since in some of our cases as seen in the Pair plot above, we had a lot of data that seemed to provide no discernible pattern. This gave us the lowest accuracy at 94.38%.

To test the rigourousness of our models, I used K-Fold Cross Validation. This uses additional samples of our data to "estimate the skill" of our models. It leads to less bias and gives a quick summary with model evaluation scores.

Overall, we have the following cross validation values.

	CV Mean	Std
Linear Svm	0.94625	0.045775
Logistic Regression	0.94375	0.045843
KNN	0.92750	0.063443
Random Forest	0.93250	0.060260



References

1. https://www.kaggle.com/terminus7/pokemon-challenge?select=combats.csv

2. https://www.kaggle.com/terminus7/pokemon-challenge?select=pokemon.csv

3. https://machinelearningmastery.com/k-fold-cross-validation/

4. https://builtin.com/data-science/random-forest-algorithm

5. https://towardsdatascience.com/predicting-cancer-with-logistic-regression-in-python

6. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

7. https://scikit-learn.org/stable/modules/svm.html