Predicting Pokémon Types with Clustering and Classification

University of Toronto - STA2201 - Winter 2025

Justin Zhang, Isaac Baguisa, Alex Faassen

2025-04-04

0.1 Abstract

- Report Summary
- Link GitHub?

0.2 Contributions

- Justin Zhang:
- Isaac Baguisa: Stats dimension reduction, Unsupervised models
- Alex Faassen: Data pre-processing, Data exploration, Image dimension reduction

1 Introduction

Over the past two decades, Pokémon has evolved from a niche Gameboy game into a global multimedia franchise encompassing anime, trading cards, e-sports competitions, and mobile applications. For many who grew up in the late 2000s and early 2010s, the world of Pokémon was a formative part of childhood entertainment alongside cultural staples like Mario Bros. and Zelda. As Pokémon continues to captivate audiences across generations, its community-oriented game design and well-structured universe presents an intriguing opportunity for data analysis. In this project, we investigate whether a Pokémon's type—a categorical label used in-game to denote elemental or behavioral characteristics such as Fire, Water, or Grass—can be inferred from its visual appearance and numerical attributes using statistical learning techniques.

Our primary research question is: *Can clustering and classification methods uncover or predict a Pokémon's type based on its image and statstical features?*

The Pokémon type system serves not only as a game mechanic but also as a conceptual grouping based on traits like colour, strength, and theme. We aim to explore whether these groupings have an underlying statistical structure that can be detected through dimensionality reduction, clustering, and classification algorithms. This analysis will provide insight into the extent to which a Pokémon's type is reflected in its appearance and physical characteristics, or whether it is a more arbitrary design choice by game developers.

2 Data Description

Our analysis is based on two publicly available Kaggle datasets:

2.1 Pokémon Image Dataset

URL: Pokémon Image Dataset

The Pokémon Image Dataset consists of 809 PNG files representing unique images of Pokémon from generations 1 through 7. Each file consists of a 3-dimensional array of dimensions 120 by 120 by 4. The first 2 slices

represent a 120 by 120 grid of pixels, and the 3rd slice represents the 4 RGBA channels (Red, Blue, Green, Alpha), the colour and transparency assignments of each pixel.

Figure 1: Images of Pikachu (Primary type: Electric) and Charizard (Primary Type: Charizard) from the Pokémon Image Dataset.

2.2 Pokémon Stats Dataset

URL: The Complete Pokémon Dataset

This structured dataset contains 41 variables for 801 Pokémon. These features include:

- **Physical attributes:** height, weight, base experience
- **Combat statistics:** HP (Health Points), attack, defense, special attack, special defense, speed
- **Categorical indicators:** Legendary status, abilities, and generation
- **Primary and secondary type labels**

All variables are numeric or have been encoded numerically (e.g., legendary status as 0/1). There are no missing values.

Table 1: Summary Statistics for Key Features

Feature	mean	sd	min	q1	median	q3	max
hp	68.96	26.58	1.0	50.0	65.0	80.0	255.0
attack	77.86	32.16	5.0	55.0	75.0	100.0	185.0
defense	73.01	30.77	5.0	50.0	70.0	90.0	230.0
sp_attack	71.31	32.35	10.0	45.0	65.0	91.0	194.0
sp_defense	70.91	27.94	20.0	50.0	66.0	90.0	230.0
speed	66.33	28.91	5.0	45.0	65.0	85.0	180.0
height_m	1.16	1.08	0.1	0.6	1.0	1.5	14.5
weight_kg	61.38	109.35	0.1	9.0	27.3	64.8	999.9

Table 1 provides an overview of the distribution of key numeric features across Pokémon in the dataset. Several notable patterns emerge:

Combat statistics, such as `attack`, `defense`, `sp_attack`, `sp_defense`, and `speed`, show similar ranges, with means around 66–78 and standard deviations near 30. These distributions suggest balanced but varied combat capabilities, with maximum values exceeding 180 or 230, likely representing fully evolved or legendary Pokémon. `hp` follows a similar pattern with a slightly lower mean of 69 and a maximum of 255, again suggesting outliers with extreme values. In contrast, the physical attributes, `height_m` and `weight_kg`, seem to be skewed. The average height is just over 1 meter, but with a maximum of 14.5 meters. Weight displays even greater disparity, ranging from under 1 kg to nearly 1000 kg. This indicates that a few extremely large Pokémon (e.g., Wailord) heavily influence these distributions.

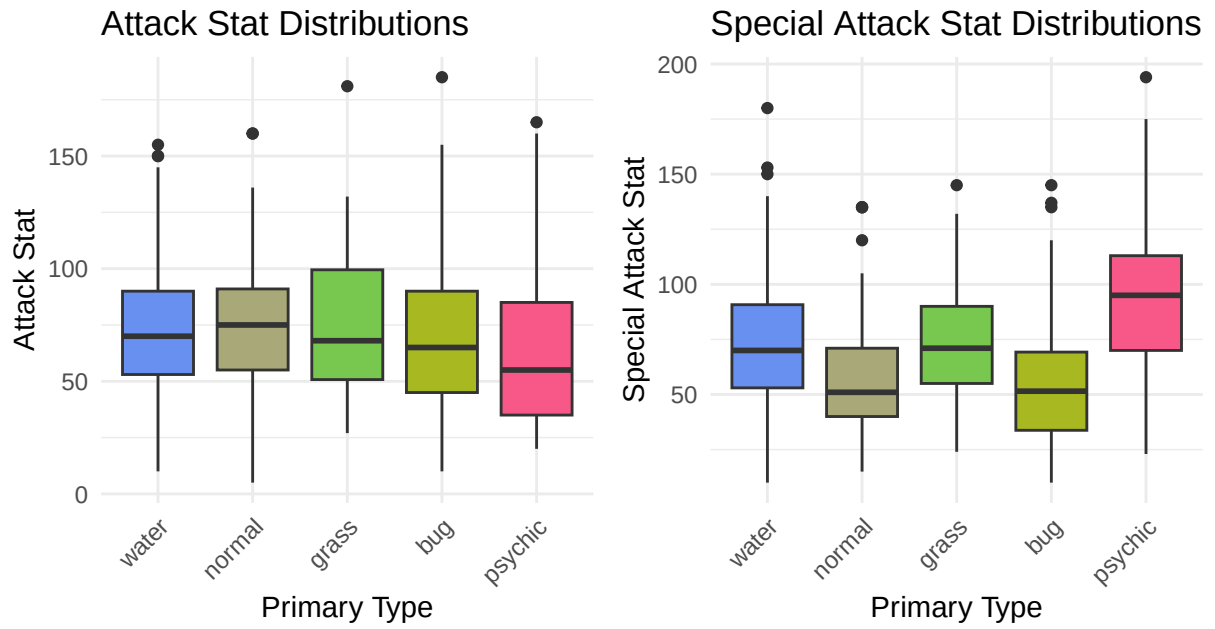


Figure 2: Frequency and attack statistic distributions of the top 5 Pokémon primary types by frequency.

In addition to summary statistics, **Figure 2** visualizes the distributions of the `attack` and `sp_attack` stats across the five most common primary types using boxplots. These five types, `water`, `normal`, `grass`, `bug`, and `psychic`, were selected based on their overall frequency in the dataset.

The boxplots reveal several key insights:

- `normal` Pokémon tend to have a slightly higher median attack, but slightly lower median special attack compared to the other types. The interquartile ranges are also both narrower, indicating a more concentrated spread of both stats within this type.
- `bug` and `psychic` types show the lowest median attack values among the five. However, `psychic` seems to make up for this with a significantly higher special attack spread, while `bug` maintains the lowest special attack stats.
- All types exhibit high-attack outliers above 150, indicating that even typically lower physically offense types can include powerful exceptions.
- The two most common types, `water` and `normal`, show narrower distributions for both stats, consistent with their broad representation across Pokémon species.

This visualization supports the idea that while some types, like `normal`, may lean toward stronger physical offense, others, like `psychic`, may rely on special attack, while some, like `bug`, may rely on `abilities` or other strategic capabilities not captured by standard combat stats alone. These findings underscore the importance of incorporating multiple features when attempting to predict a Pokémon's type using clustering or classification models.

2.3 Pre-processing

For the images, we begin by flattening each PNG into a (120x120x4) 57600-element vector, each pixel and RGBA value representing a feature. Binding these vectors into a table identified by Pokémon name gives us a usable dataset. To integrate image features with the stats dataset, we filter both tables to common Pokémon requiring several name formatting adjustments to account for unusual characters. Finally, we match the row-order of both datasets, resulting in two easily transferable Pokémon datasets.

2.4 Primary Types

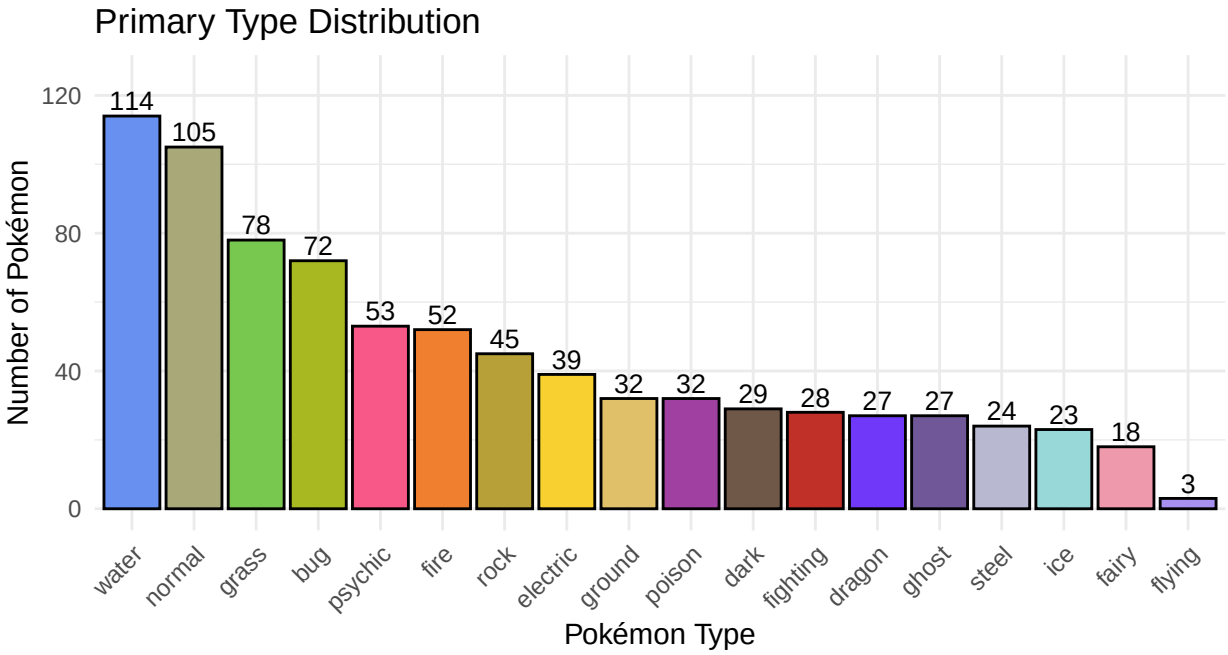


Figure 3: Pokémon primary type distribution.

Figure 3 shows the distribution of Pokémon by the **18 primary types** across the transformed datasets. For this project, we **only consider primary type** labels to maintain a single-label classification structure. With this in mind, we observe a notable class imbalance, with certain types like `water` (114 Pokémon), `normal` (105), and `grass` (78) appearing much more frequently than rarer types such as `fairy` (18) and `flying` (3). This uneven distribution has important implications for clustering classification metrics and interpretations.

3 Methodology and Results

In this section we run three types of analysis: principal component analysis, unsupervised learning (k-means), and supervised learning (linear discriminant analysis, gradient boosting). We will describe the analysis conducted and then showcase the interesting results. Note that non-informative results will be in the appendix or omitted for brevity.

3.1 Image Dimension Reduction

To explore whether Pokémon types are distinguishable based on their visual features, we look to apply clustering algorithms to their images. Since we are working with high-dimensional data (801x57600), we first apply dimension reduction techniques to mitigate the curse of dimensionality. Our primary tool for image dimension reduction is Principal Component Analysis (PCA). In its own right, this method also helps us determine if type-based structure is embedded in high-dimensional image representations.

3.1.1 Principal Component Analysis (PCA)

PCA is a linear technique that identifies the directions of maximal variance in the dataset. After centering the flattened image matrix, we applied PCA and examined the principal components, loading vectors, and compression options.

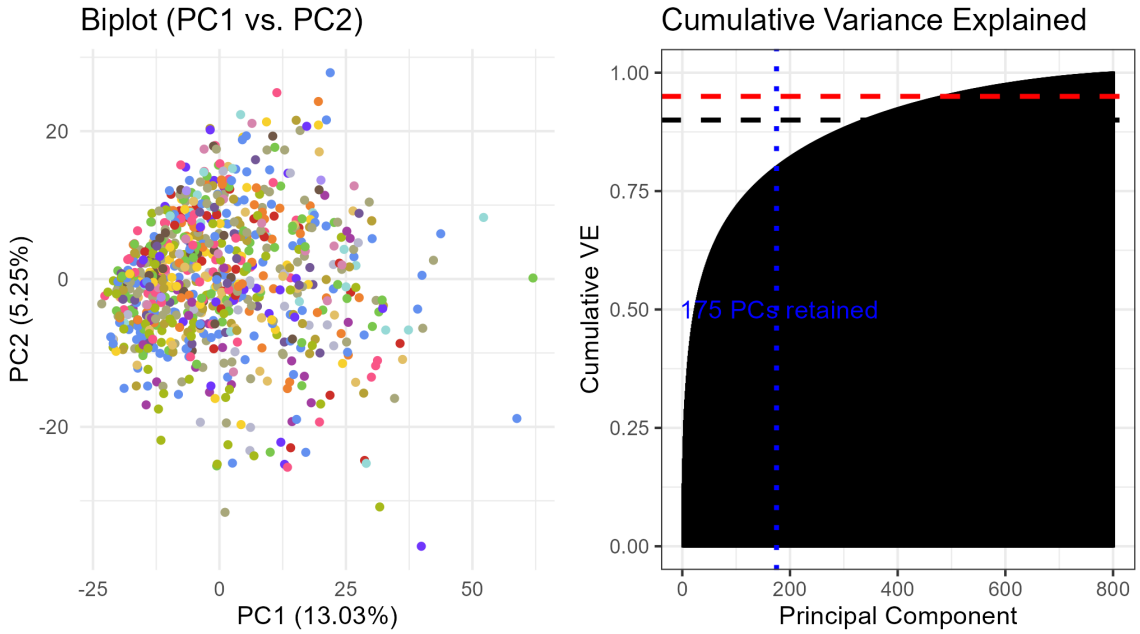


Figure 4: Left: Cumulative Variance Explained (VE) by principal components (PC). The blue and black dashed lines represent 80% and 95% of VE respectively. Right: Biplot for the first 2 PCs.

Figure 4 above summarizes the results of applying PCA to the flattened Pokémon image data.

On the left, the biplot visualizes the Pokémon with respect to the first 2 principal components (PCs), **PC1 (13.03%)** and **PC2 (5.25%)**. Each point corresponds to a Pokémon, colour-coded by its primary type. While its possible that there are weak groupings, all types seem to be heavily overlapping. This is somewhat unsurprising given the first 2 PCs only represent roughly 18% of the variance. Further, **Appendix A** examines the first 2 **loading vectors** which seem to correspond to Pokémon area (i.e. height-width stretch) and height-width ratio respectively, neither of which seem particularly promising on their own for type clustering. These findings motivate the use of non-linear methods, such as **UMAP** or **t-SNE**, which are better suited for uncovering type-based structure in complex image data.

On the right, the **cumulative variance explained** plot shows that the first few PCs capture a substantial portion of the variance, but with diminishing returns beyond the first 100 components. To retain at least **80% of the total variance**, we selected the first **175 PCs** (indicated by the blue vertical line). This threshold strikes a balance between compression and information preservation with a slightly greater emphasis on compression, and these 175 components are used in downstream clustering and classification steps. Below, **Figure 5** and **Appendix B** provide visual examples of the degree of compression applied before applying clustering algorithms.

3.1.2 UMAP

To uncover non-linear structure in the image data, we applied **Uniform Manifold Approximation and Projection (UMAP)**. Unlike PCA, which is a linear method, UMAP is designed to preserve both local and global relationships in high-dimensional spaces, making it particularly effective for image-based clustering.

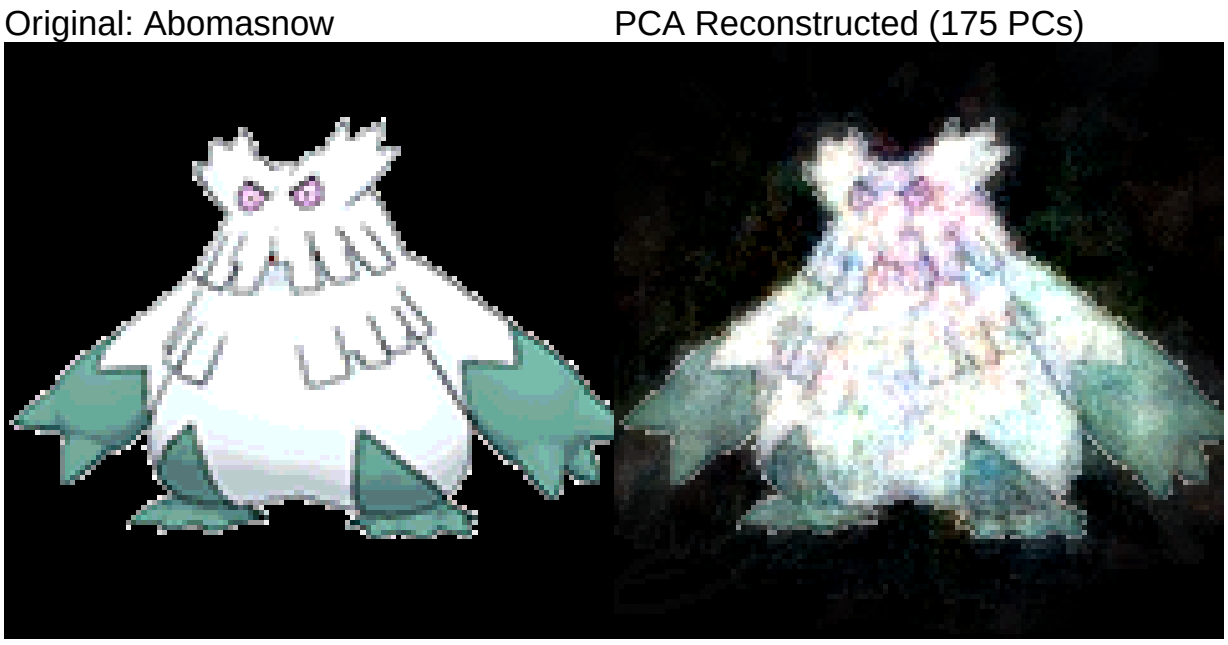


Figure 5: Images of the Pokémon Abomasnow. Left: Original. Right: After image compression with 175 PCs, capturing 80% of the variance explained.

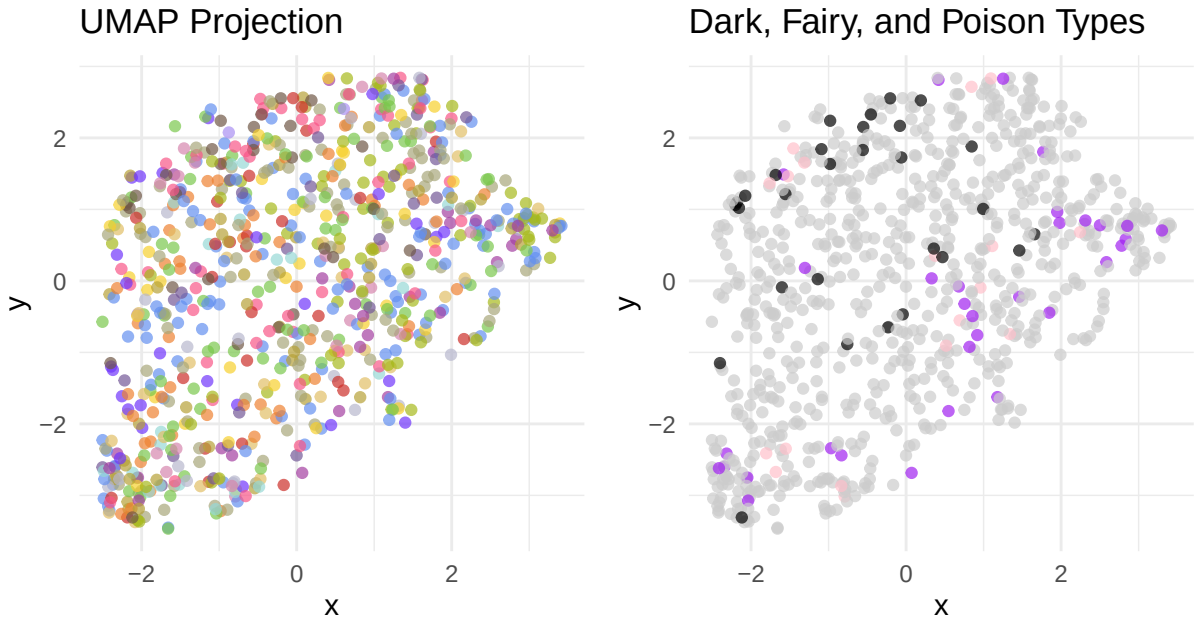


Figure 6: 2D UMAP projection of the 801 Pokémon image vectors. Left: Coloured by their primary type. Right: All type colours set to grey, save Dark, Fairy, and Poison types.

Figure 6 displays the 2D UMAP projection of the 801 Pokémon image vectors, coloured by their primary type.

Once again, with the large numper of types, it is very hard to visually identify groupings. Upon close inspection, there does seem to be slightly more grouping between certain types, such as `dark`, `fairy`, and `poison`. However, for the most part, most types appear to be largely overlapping. Though types aren't particularly distinct, there does seem to be some improvement over the PCA biplot. Overall, UMAP provides stronger visual evidence that type-specific patterns may exist in the high-dimensional image data.

3.2 Unsupervised Model

This section evaluates if unsupervised models such as K-means clustering can accurately recover Pokemon types based on their numeric stats and image features. Specifically, we explore standard K-means clustering, K-means++ initialization, and weighted K-means clustering methods (distance-based weighting). We used dimensionality reduction methods such as PCA and UMAP before clustering the image data. We examine two Pokemon datasets, one including numeric stats (attack, defense, speed) and one including image data. Numeric missing data was imputed using K-Nearest Neighbours (KNN) imputation, and features with zero variance were removed. PCA was applied to both numeric data and image data to reduce dimensionality. For the numeric stats dataset, 20 principal components that explained 90% of the variance were kept. We also applied clustering methods on the UMAP features from the images to compare its accuracy against PCA.

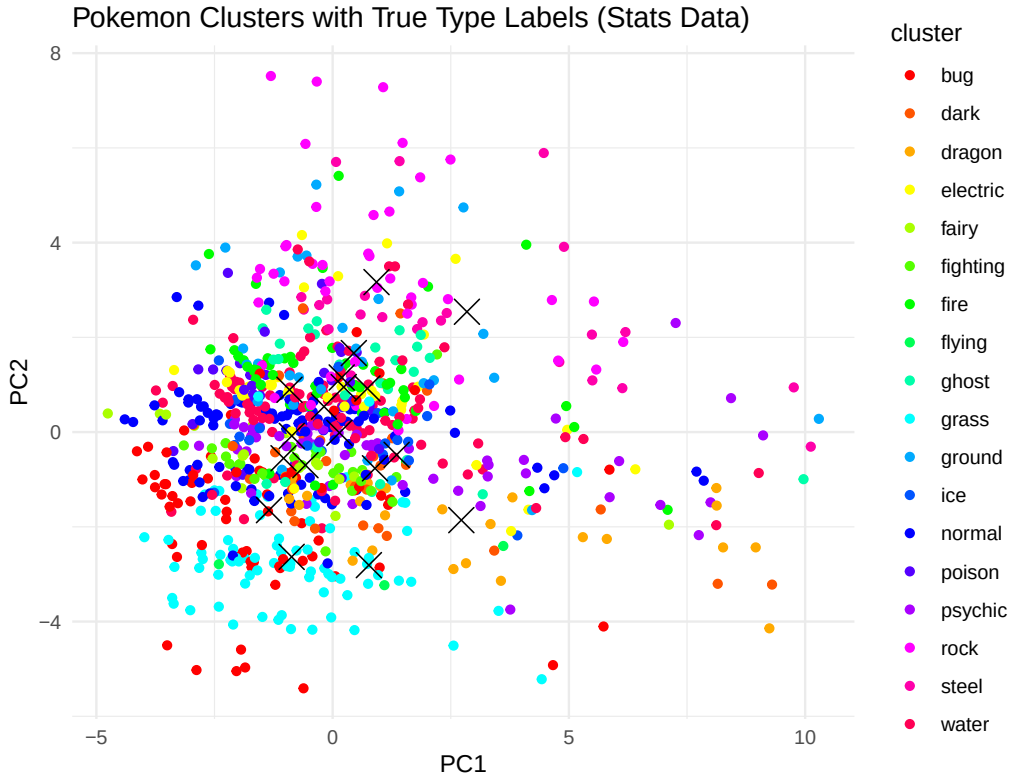


Figure 7: Visualization of true Pokemon types on PC1 and PC2

We set the initial number of clusters (k=18) to match the total number of Pokemon types. The clustering methods were compared using within sum of squares (wss), between sum of squares (bss), CH index, and accuracy. The labels of each cluster was assigned by labelling each cluster the most common type in that cluster - this allows us to compute accuracy by comparing the assigned labels with the true types.

Clustering performance was also assessed by determining alternative optimal cluster numbers, identified through the elbow method (wss), silhouette score, CH index, and gap statistics. Testing for different number

of clusters may help identify if there are consistent trends with the grouping of certain types of Pokemon, such as if there are certain types consistently grouped together. Additionally, if the optimal number of clusters is small, it suggests that the data might naturally form fewer larger groups. Comparing standard K-means and K-means++, the clusters had a wss of 12613.8 and 12848.89 respectively, and bss of 14360.22 and 14125.13 respectively. This suggests the stats dataset creates more distinct clusters with standard K-means clustering.

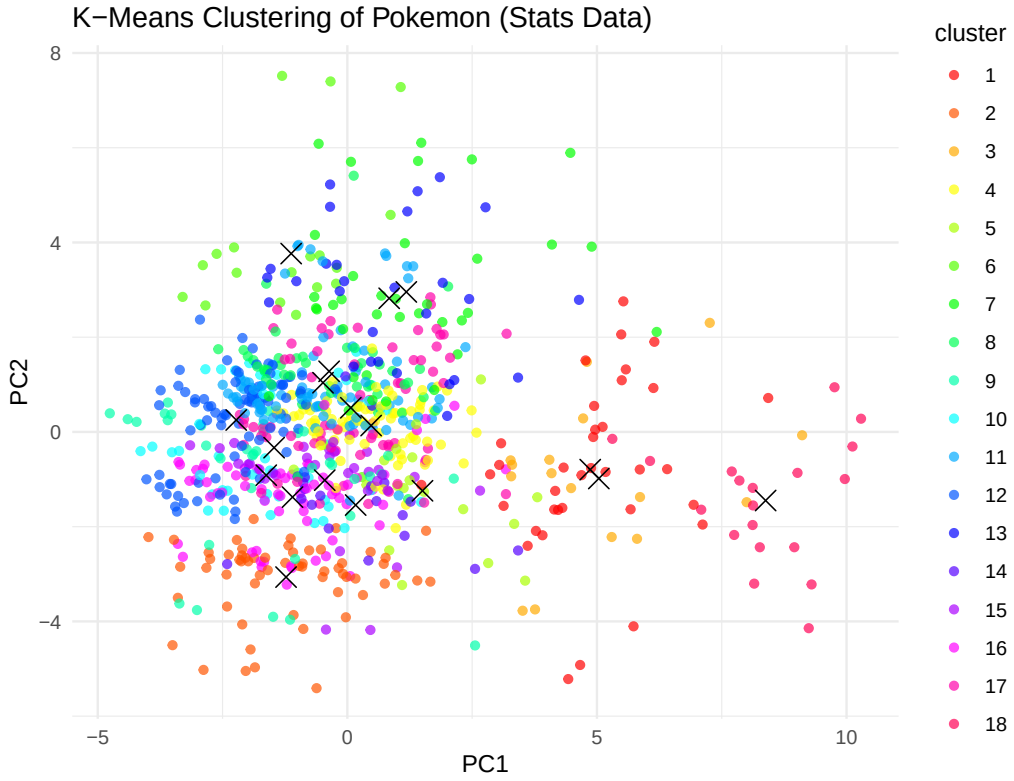


Figure 8: Visualization of Pokemon types from K-means clustering on PC1 and PC2

The clustering performance on numeric Pokemon stats resulted in moderate accuracy, with standard K-means and K-means++ methods performing similarly (approximately 52.4% - 52.9% accuracy). Weighted K-means had much lower accuracy (20.3%), suggesting that the distance based weighting did not benefit clustering with these data.

Table 2: Clustering Method Accuracies on Stats Data

Method	Accuracy
K-means	0.5243446
K-means++	0.5293383
Weighted K-means	0.2034956
Optimal K-means K = 3	0.1722846

Optimal cluster methods resulted in notably fewer clusters (ranging from k = 1 to 10) than the actual number of Pokemon types (18). The elbow method suggested k = 3, silhouette suggested k = 2, CH index suggested k = 1, and gap statistics suggested k = 10. Using k = 3, the accuracy was 17.2%, which is moderate as the types remaining were grass, water and psychic, which make up about 30.5% of the dataset. Overall, these results indicates that Pokemon numeric stats do not naturally cluster into distinct groups corresponding directly to each of the 18 types.

In the image dataset, comparing standard K-means and K-means++, the clusters had a wss of 677488.2 and 696824.9 respectively, and bss of 288142.1 and 268805.4 respectively. This suggests the image dataset creates more distinct clusters with standard K-means clustering.

Image clusters consistently produced lower accuracy than numeric stats. Standard K-means had approximately 18.1% accuracy, while K-means++ and weighted K-means performed similarly or worse (17.4% and 14.2%, respectively). This implies image features alone were insufficient to recover Pokemon types.

Optimal cluster tuning again produced fewer optimal clusters (k = 1 to 10). Using k = 3, accuracy intuitively dropped further to 14.7%, indicating images likely contain common and general features across many Pokemon types, such as similar color or shapes, leading to ineffective cluster separations. The poor clustering accuracy highlights significant weaknesses in the image-based approach. Pokemon image characteristics appear not to strongly correlate with type when reduced to PCA features.

Table 3: Clustering Method Accuracies on Image Data

Method	Accuracy
K-means	0.1810237
K-means++	0.1747815
Weighted K-means	0.1423221
Optimal K-means K = 3	0.1473159

3.2.1 UMAP-based Clustering

UMAP dimensionality reduction was also applied on the image data to see if non-linear dimension reduction improves accuracy for recovering types. K-means with UMAP gave 18.6% accuracy and K-means++ with UMAP gave 18.2% accuracy. Although UMAP aims to uncover non-linear patterns, accuracy remained consistently low. UMAP's nonlinear features did not significantly improve the separation of Pokemon types compared to PCA. This further confirms the difficulty in extracting type-specific visual characteristics from Pokemon image data, reinforcing the idea that image features do not map effectively onto the Pokemon types.

3.2.2 Comparing Clustering Methods

Standard K-means and K-means++ gave similar performances across numeric and image datasets. Standard K-means slightly outperformed K-means++ in numeric stats, whereas K-means++ was less effective on image data. Weighted K-means consistently underperformed, which suggests that distance-based weighting do not match the structures in these datasets. Numeric stats clustering significantly outperformed image clustering, indicating that numeric stats more closely align with Pokemon types than images.

3.2.3 Standard K-Means (K=18) Interpretations

Looking deeper into the clustering results from standard K-means with k=18, we see how the K-means algorithm groups Pokemon based on their stats, independent of their actual types. This gives us an idea of whether Pokemon types with similar stats (high attack and speed, or high defense and hp) tend to cluster together. In general, it is common to see Pokemon with relatively similar attack, speed, or defensive stats cluster together, but this does not necessarily correspond to the type the Pokemon belongs to (See Table 4 in Appendix). In Table 5 in the appendix, some clusters were clearly dominated by a single type - such as water for cluster 11. There were some common clusters containing a mix of different types that could be linked through common attributes. For example, cluster 17 had ghost and psychic types dominate in this cluster - suggesting these types share common statistical features. Similarly, cluster 10 grouped many bug, grass, and poison types. Overall this suggests that statistical features like attack, defense, etc., might outweigh type-specific features when forming clusters.

3.3 Supervised Model

This section explores the performance of supervised models on the Pokemon stats dataset (note that results for the Pokemon images dataset is not presented as they provide no inferential capabilities). The purpose is twofold: to provide a baseline accuracy benchmark to compare the unsupervised models to, and to help determine whether we can predict Pokemon type. To do this we will run two models, linear discriminant analysis and gradient boosting. Linear discriminant analysis uses Bayes classifier to determine linear decision boundaries between types. As such, it is flexible to outliers, of which there are quite a few in our datasets, though it naturally performs worse when decision boundaries are non-linear, which is also the case. Gradient boosting is (at a high level) an ensemble learning algorithm based that iteratively fits decision trees on the previous residuals and updates prediction. In our classification setting, we use the log odds as the response and run the model for each class at each iteration. The major benefit here is that gradient boosting works quite well still with non-linear, non-separable data, though the downside is overfitting (which we can mitigate with hyperparameter tuning). The models are run on the same PCA dimension-reduced data as in the clustering section. Between the two models, we hope to learn the statistical makeup up different types of Pokemon.

In these supervised models, we use 80-20 stratified train-test splits. This means within each Pokemon type, we choose 80% of the Pokemon to be in training set, and remaining 20% in test set. This will prevent over representation of certain types in each split, which will effect prediction accuracy for that specific type. This is particularly prevelant when we sub-group by generation, which results in small overall numbers of each Pokemon type.

3.4 Linear Discriminant Analysis

We start by running LDA on the entire PCA-reduced Pokemon stats dataset. The prior probabilities used are the types proportions. Results are presented in Figure 8 and Table 4. Our two-dimensional plots show a lack of linear separability, however incorporating all dimensions, LDA does a good job of classification, as evidence by the training accuracy hovering around 88%. Test accuracy is essentially the same as train accuracy, showing the model does not suffer from overfitting, generalizing very well. We run reduced-rank LDA as a form of regularization, however there are no improvements to test accuracy by using low-dimensions. This can be partially explained by the fact we have already applied PCA to the original data, which itself is a rank-reduction technique.

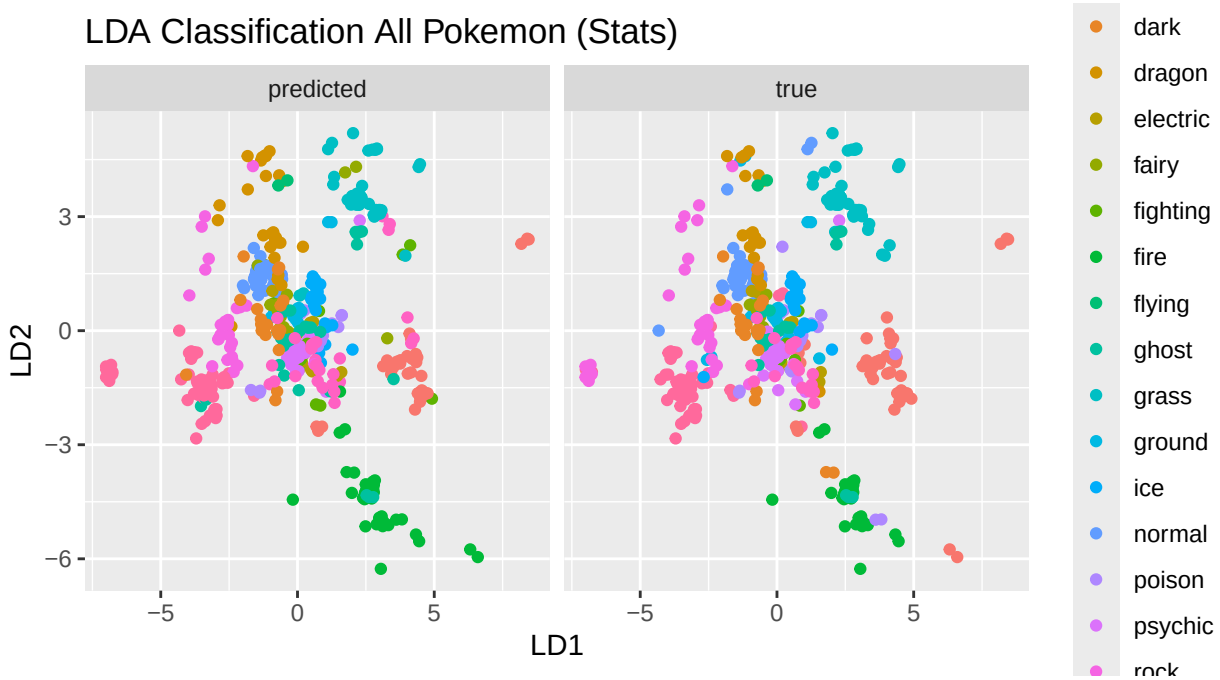


Table 4: LDA Accuracy for Stats Data

L	accuracy_train	accuracy_test
1	0.4574132	0.4371257
2	0.6167192	0.5808383
3	0.7176656	0.6886228
4	0.8091483	0.7904192
5	0.8454259	0.8263473
6	0.8517350	0.8383234
7	0.8738170	0.8622754
8	0.8864353	0.8862275

We next run LDA individually on each generation of Pokemon. Results are presented in Figure 9 for generation 1 and Table 5 for generation 1 to 4. Visually, looking at the first 2 components, LDA is able to cleanly separate types with linear boundaries. All 20 components are almost entirely linearly separable by hyperplanes. Accordingly the train accuracy is at worst 95% in each generation, meaning near perfect classification. We do however see that test accuracy is lower by on average 11% for each generation, meaning that there is some overfitting.

From this we can conclude that the generation of the Pokemon generation is a blocking variable in the classification of type, and it is better to run models on each individual generatio. This can partly be explained by the non-standardization of both power level (ex. health, attack) and type across generations due to terrain (in the Pokemon world), number of legendaries etc.

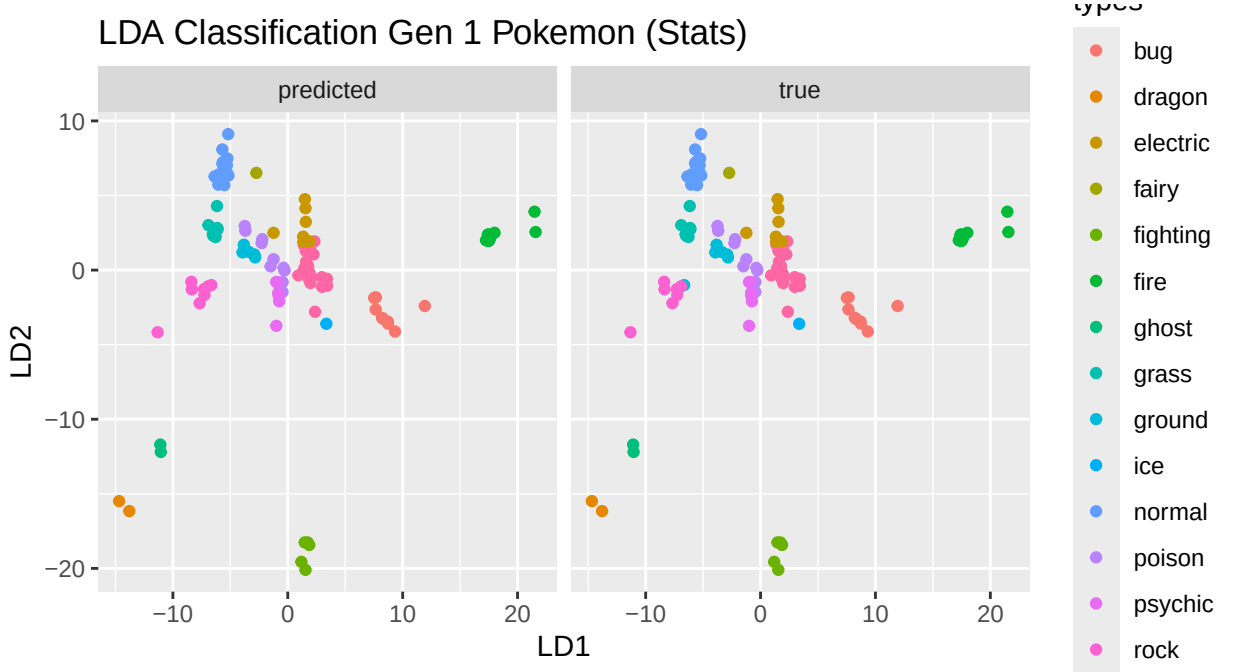


Table 5: LDA Accuracy for Stats Data - by Generation

	gen1-train	gen2-train	gen3-train	gen4-train	gen1-test	gen2-test	gen3-test	gen4-test
3	0.9824561	0.9459459	0.970297	0.9480519	0.9189189	0.6923077	0.7647059	0.5333333
4	0.9912281	1.0000000	0.960396	0.9870130	0.9189189	0.7692308	0.8235294	0.6000000
5	0.9912281	1.0000000	0.970297	0.9870130	0.9189189	0.8076923	0.8529412	0.6333333
6	0.9912281	1.0000000	0.970297	1.0000000	0.9189189	0.8076923	0.8235294	0.6333333

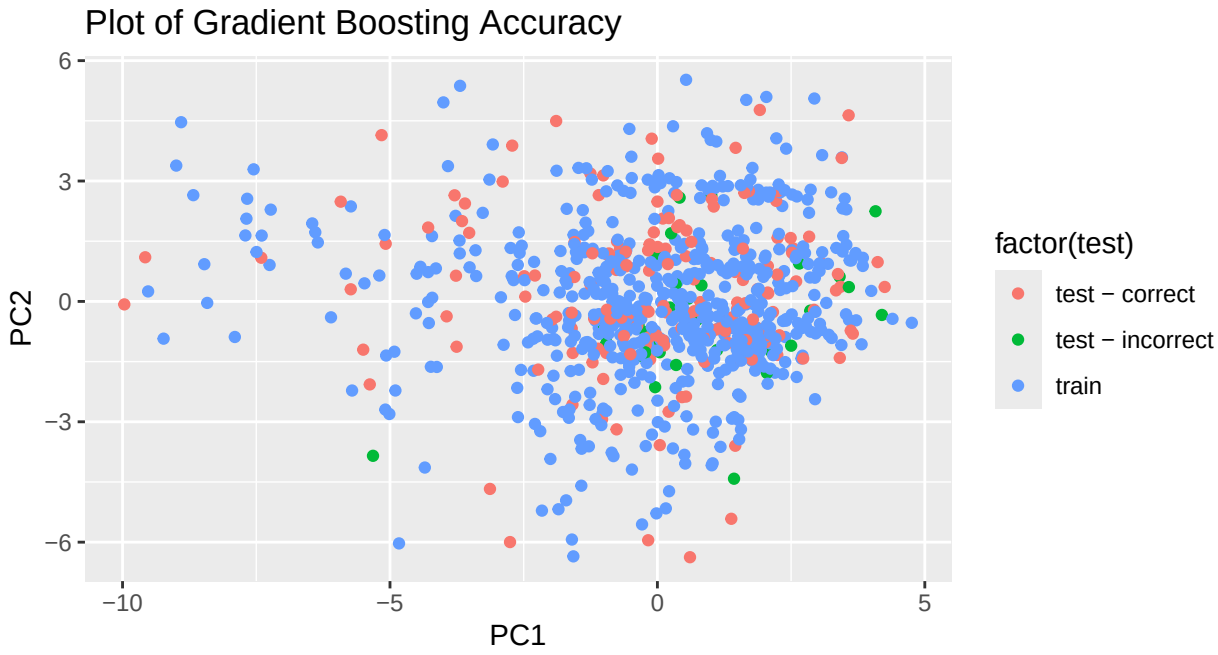
3.5 Gradient Boosting

We run gradient boosting (XGBoost) on the probability of each Pokemon being a specific type. Results of these models are displayed in Table 6. To keep this from becoming a hyperparameter tuning exercise, we used the values 100 iterations (trees), max depth of 3, while tuning for shrinkage factor, which is optimal around 0.2. The training accuracy is near perfect, however, we do see loss of accuracy on the test, which is consistent with our knowledge that gradient does tend to overfit data. Figure 10 shows which Pokemon in the test set were missclassifed. We also ran the gradient boosting on each specific generation, and again test accuracy noticeably improves.

These results show that LDA is the superior statistical method in predicting Pokemon type. This is due to its flexibility in model fit, as gradient boosting suffers from overfitting. Overall, these results show that we are able to predict Pokemon type based on statistics with high accuracy.

Table 6: Gradient Boosting Accuracy

shrinkage	train_accuracy	test_accuracy
0.01	0.7271293	0.6047904
0.10	0.9968454	0.8143713
0.25	1.0000000	0.8383234



4 Discussion

- Discuss model performance
- Limitations of methodology
  - Potential sources of bias
- Challenges encountered
- Recommendations for overcoming these + improvements for future work

Overall, the models did a far better job of predicting Pokemon type than we had originally anticipated. As Pokemon enjoyers, oftentimes there is little rhyme or reason to why certain types have specific attributes, as it is Pokemon specific. We thought the structure of the data may not capture the complexity of the Pokemon types in a way that creates efficient and distinct clusters. However, clustering on Pokemon stats still

resulted in around 50% classification accuracy (across 18 classes), which though not great, beat expectations. Supervised learning algorithms had very high test accuracy, showing that classification based on Pokemon stats is quite applicable. On the other hand, classification based on Pokemon image was not successful, with low accuracy in both supervised and unsupervised settings.

A significant limitation we faced is non-uniformity between number of each Pokemon types. Notably, fairy and dragon types were not even introduced until later generations, meaning there is a comparative lack of them in the dataset. In supervised models, train-test splits used stratification to account for this issue. In unsupervised learning, it caused an inability to cluster together groups with comparatively low numbers, with most clusters being predominantly 'water' or 'normal' types. Another limitation is the presence of dual typing, meaning Pokemon have a primary and secondary type, which introduces a confounding factor. For example, fire-fighting Pokemon and fire-psychic Pokemon are quite different, which will make it more difficult to accuratley classify the Pokemon type.

Our calculation of classification accuracy in clustering is biased by the non-uniformity of Pokemon types. Since we assign each cluster prediction to the cluster mode type, types like 'water' and 'normal', which make up the bulk of the dataset will naturally be predicted more. This means that classification accuracy is inflated by not prediciting the low-number types like fairy and flying. In fact, if we ensured that each type was only predicted for a single cluster, accuracy drops sharply. Another possible source of bias is our use of the 'against_(specific type)' variables. Pokemon enforces the laws of nature, and so some types will fare very well against others (ex. water against fire). Though this is not inherently wrong to include, it does not help answer our research question, as these variables are performance metrics rather than fighting attributes or appearance. In fact, if we only use fighting attributes classification accuracy for K-means drops to 29%.

Some of these biases and limitations like non-uniformity of Pokemon types is intrinsic and hard to correct for (specifically in unsupervised models). We can account for dual typing in supervised models by classifying each hybrid type individually, though this may run into problems with low samples size for individual hyprid types. When working with Pokemon image data, we can change our approach from treating each pixel as a variable to creating our own descriptional variables from the images such as 'presence_of_(colour)' and 'number_of_non-white_pixels'. In this way we can hopefully cluster based on image characteristics, which do change by type (for example fire Pokemon often have red). Furthermore, we combine these image characteristics with the statistics dataset, and run analysis on that. Additionally, we can try out different types of models for both supervised and unsupervised learning, like Gaussian mixture models.

5 References

- Use `knitcitations`? - manual easier?

6 Appendix

6.1 A: Loading Vectors

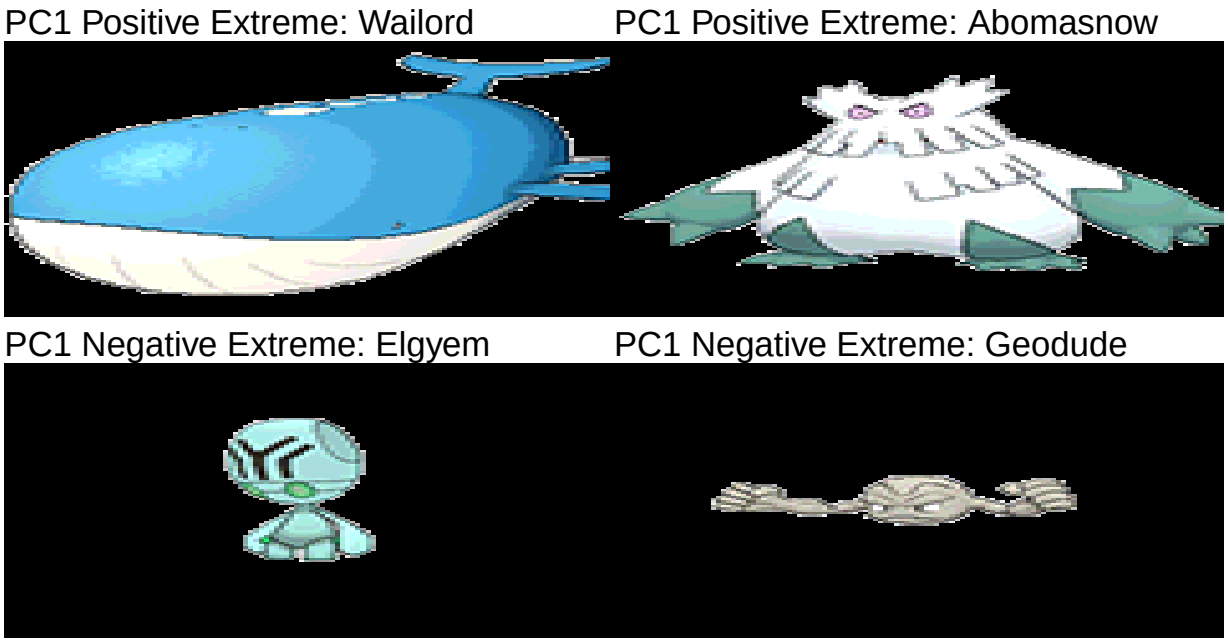


Figure 9: PCA PC1 extreme value Pokémon. Seems to reflect area.

6.2 B: Gardevoir Image Compression

6.3 C: Clustering

Table 7: Average Statistics Rating Across Standard K-means Clusters (Stats Data)

km_cluster	attack_mean	hp_mean	defense_mean	speed_mean
1	101.36111	86.36111	96.52778	92.77778
2	70.66667	64.26667	68.76667	58.51667
3	109.62500	92.68750	94.93750	109.18750
4	90.71429	87.19481	75.12987	80.46753
5	102.05263	76.63158	94.42105	78.63158
6	80.26316	60.57895	73.94737	67.89474
7	89.13889	61.41667	114.52778	53.77778
8	77.36957	64.32609	60.69565	72.76087
9	53.32432	67.05405	60.62162	55.89189
10	67.11111	59.57778	62.73333	59.64444
11	74.13636	67.50000	78.82955	59.18182
12	52.20000	49.84706	47.96471	51.68235
13	97.17241	76.82759	106.41379	42.27586
14	80.84211	59.63158	70.15789	56.42105
15	104.72727	74.66667	70.90909	65.27273
16	75.60938	64.53125	58.01562	83.18750
17	62.86301	64.60274	67.54795	61.49315
18	121.68421	120.36842	104.10526	87.10526

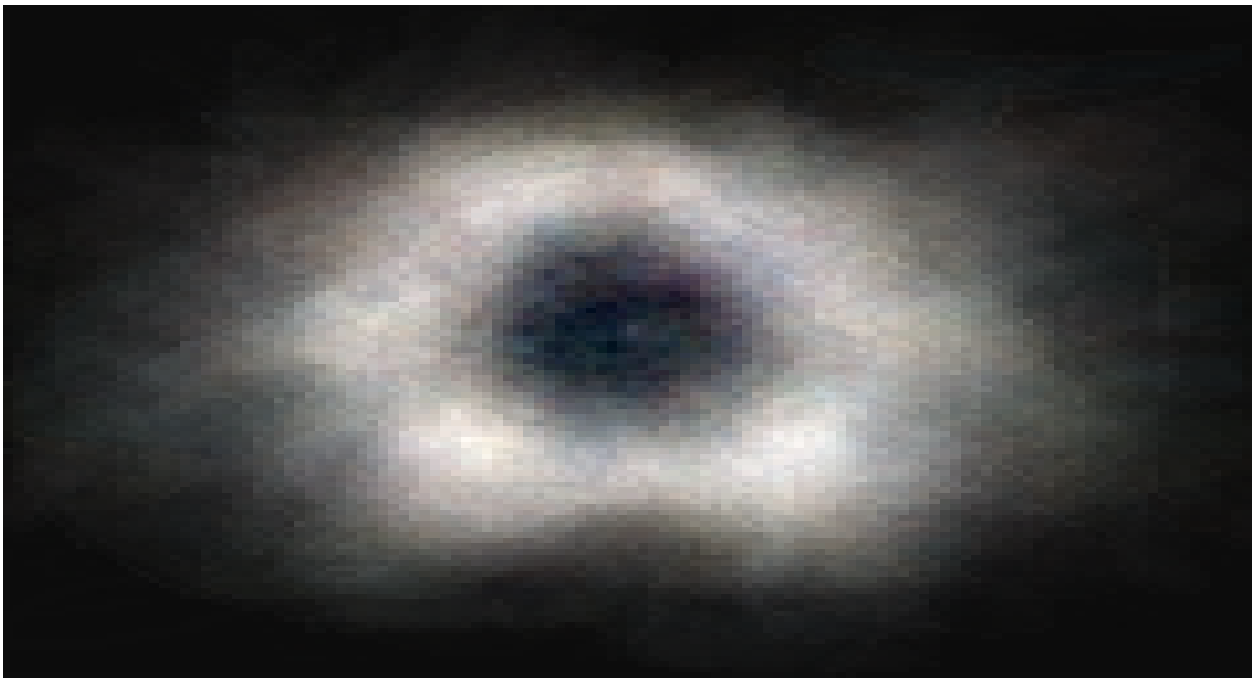


Figure 10: PCA 1st loading vector.

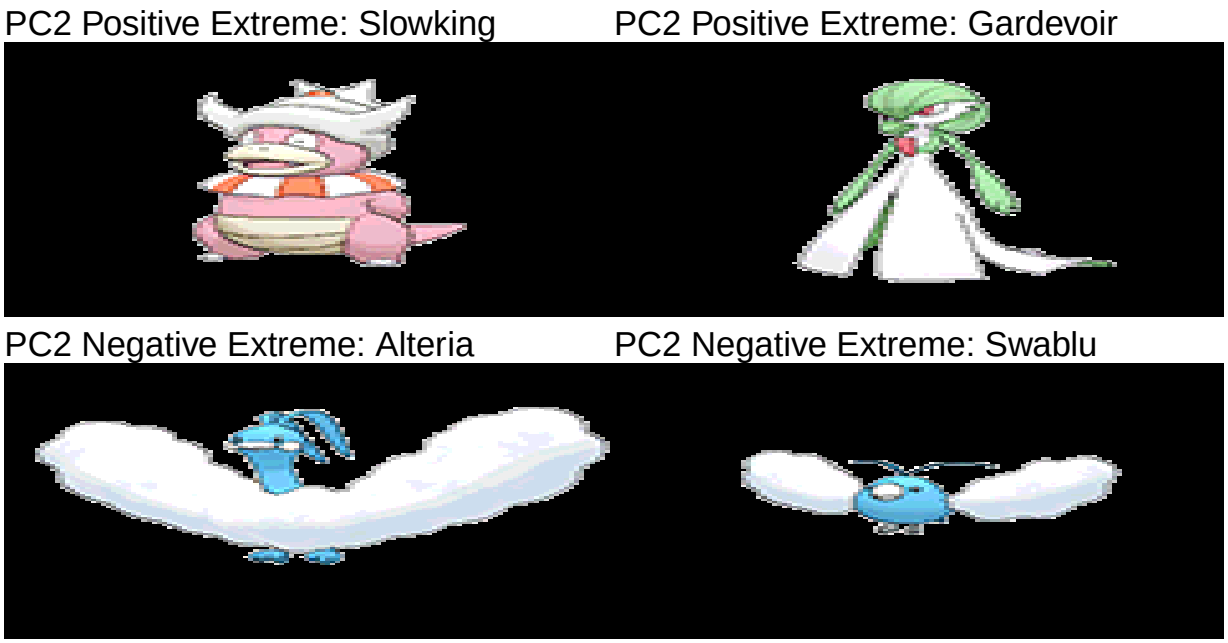


Figure 11: PCA PC2 extreme value Pokémon. Seems to reflect height-width ratio.

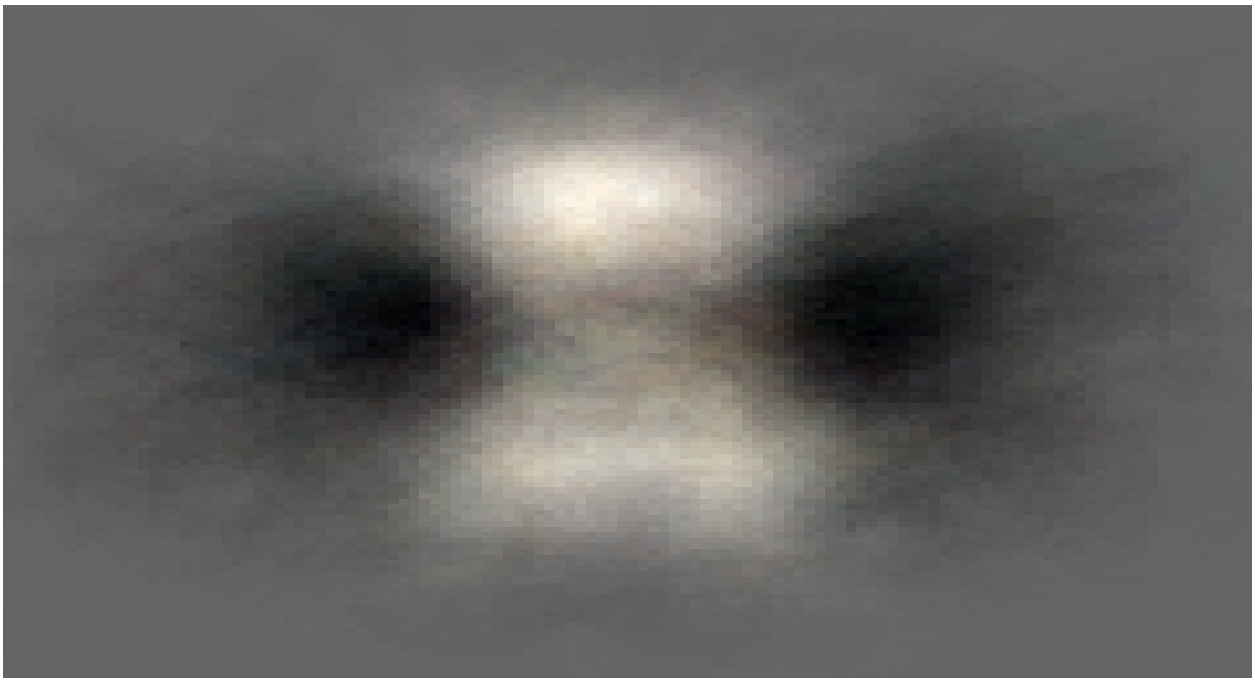


Figure 12: PCA 2nd loading vector.

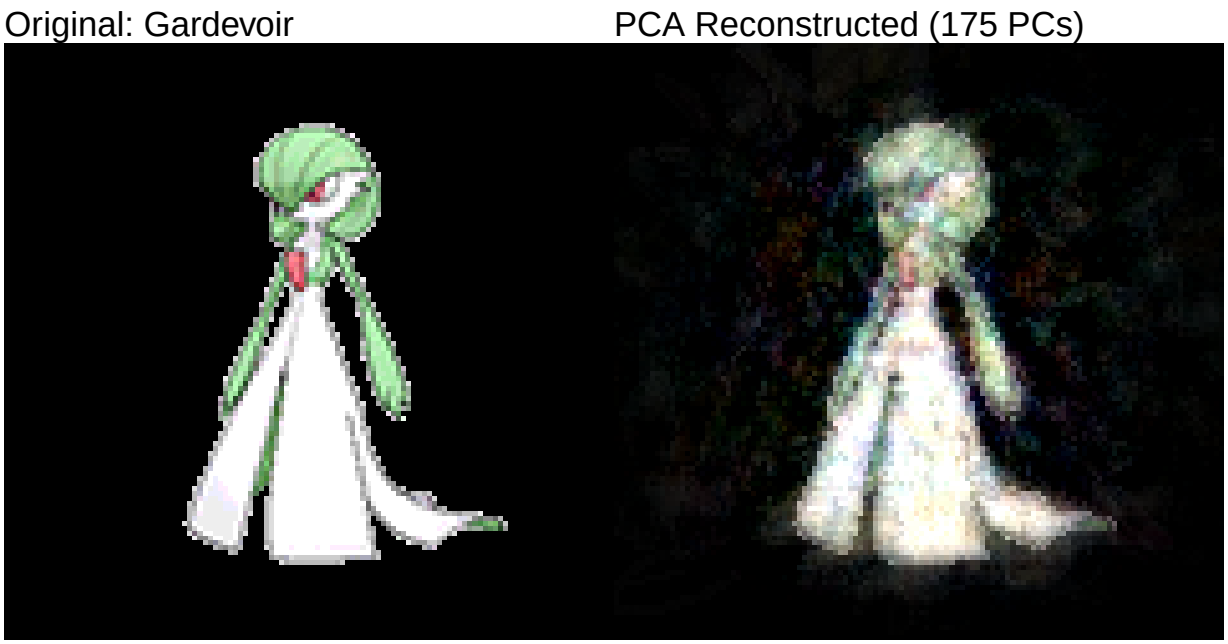


Figure 13: Images of the Pokémon Gardevoir. Left: Original. Right: After image compression with 175 PCs, capturing 80% of the variance explained.

Table 8: Summary of Cluster Assignments from Standard K-means (Stats Data)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
bug	3	6	0	3	0	0	4	2	2	12	1	19	2	0	1	16	1	0
dark	1	0	0	10	0	0	2	1	0	0	0	4	0	5	0	3	1	2
dragon	0	0	2	0	9	0	0	0	0	0	0	0	0	11	0	0	0	5
electric	5	0	0	9	0	1	4	2	1	0	0	15	0	0	0	1	1	0
fairy	1	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0
fighting	0	0	0	0	0	0	1	0	0	0	0	1	0	1	25	0	0	0
fire	3	0	0	2	1	2	1	38	0	0	0	0	0	0	0	4	0	1
flying	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
ghost	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26	1
grass	2	50	1	0	0	1	2	0	5	14	0	0	0	0	2	0	1	0
ground	1	0	0	2	4	5	1	0	0	0	0	8	6	0	0	0	4	1
ice	2	0	0	7	0	0	0	0	0	0	1	6	3	0	0	1	3	0
normal	2	2	1	38	1	4	0	0	4	0	1	22	0	0	2	26	0	2
poison	0	0	0	2	1	2	0	3	0	17	3	1	0	0	1	2	0	0
psychic	4	0	11	0	0	0	0	0	5	0	0	0	0	0	1	0	30	2
rock	4	0	0	0	1	4	3	0	0	0	6	2	18	1	0	4	2	0
steel	3	0	1	0	0	0	17	0	0	0	0	0	0	0	0	0	1	2
water	4	2	0	4	1	0	1	0	3	2	76	7	0	0	1	7	3	3