Data Analyses on Pokémon Data

Daniel Pybus
p4261837@live.tees.ac.uk

James Corcoran
j9049758@live.tees.ac.uk

Leo Chun Yu Ng
w9190570@live.tees.ac.uk

Vincent Yu Shing Lui
a0201746@live.tees.ac.uk

Bibi Kurian
w9457273@live.tees.ac.uk

Bola Odunola
a0141109@live.tees.ac.uk

Keywords: Clustering, Classification, Pokémon

Abstract

Aiming to provide information for Pokémon game players, data analyses on a Pokémon dataset were performed. Explorative K-means and hierarchical clustering were attempted under various settings and a three clusters model was finally proposed to group Pokémons based on their characteristics. Classification system was developed to identify Legendary Pokémons by their basic stats. Lastly, strongest and weakest types of Pokémon were found using correlation and clustering. Implications, uses of results, and possible future studies were discussed.

Introduction

"Pokémon" is an electronic game series developed by Nintendo where players catch, raise and combat with fictional creatures called "Pokémon". Data analysis studies were done on Pokémon data. Clustering was used to define groups of Pokémons by basic stats in study 1. In study 2, Legendary Pokémons were identified utilising Classification and Regression Trees (CART) method. Study 3 attempted to deduce the strongest and weakest types of Pokémon by correlation and clustering. These analyses aim to provide information to players beneficial to their gameplay.

Description of dataset

This open-source dataset from Kaggle contains the list of the 890 Pokémons (1028 including varieties) until 8th Generation. It includes Pokédex number, name, types, and some stats etc. For this project, we will mainly focus on several attributes as below:

1. HP: the energy of a Pokémon can withstand before faint
2. Attack: the attack value of a Pokémon fights the others (eg. Scratch, Punch)
3. Defense: the defense value to resistance against a Pokémon attack
4. SP Attack: special attack, the value of special attacks ability (e.g. fire blast, bubble beam)
5. SP Defense: special defense, the value of special defense resistance against special attacks
6. Speed: determines which Pokémon attacks first each round
7. Height: the height of each Pokémon
8. Weight: the weight of each Pokémon
9. Status: whether a Pokémon is Normal, Legendary, Sub-legendary or Mythical

(Data Source: https://www.kaggle.com/mariotormo/complete-Pokémon-dataset-updated-090420?select=pokedex_%28Update_05.20%29.csv)

1. Study 1: Clustering on Pokémon dataset (Leo & Vincent)

1.1. Objective

The main purpose of this study is to explore possible grouping of Pokémons by their stats with clustering. Characteristics or strengths of different groups of Pokémons were indicated by their stats. Attempt to compare the new grouping with existing categories was also done. Lastly, the best model to group the Pokémons was defined.

1.2. Methods

1.2.1. Attributes

Six scenarios with different combinations of attributes shown below were investigated.

Scenarios	HP	Attack	Defense	SP Attack	SP Defense	Speed	Height	Weight	Number of attributes used
A	✓	✓	✓	✓	✓	✓	✓	✓	8
B	✓	✓	✓	✓	✓	✓			6
C	✓	✓	✓			✓			4
D	✓	✓	✓						3
E							✓	✓	2
F				✓	✓				2

1.2.2. Data preparation

```
# Read data from csv
pokemon_all <-read.csv("C:/MSC/pokedex.csv")
# Select relevant variables
pokemon_stats <- select(pokemon_all, height_m, weight_kg, hp, attack,
defense, sp_attack, sp_defense, speed)
#Standardized Z-scores
pokemon_z <- as.data.frame(scale(pokemon_stats, center = TRUE, scale =
TRUE))
```

Missing data were first explored. Due to minimal number of missing data, and no proper substitution applies, cases with missing data in any attributes were deleted from analyses.

Data transformation was necessary due to different ranges and distributions in attributes (see descriptive statistics in Table 1). Standardization was used to rescale values in each attribute into Z-score by subtracting the mean from the value and dividing the result by the standard deviation. The result transforms set of values which has standard deviation of one and mean of zero (Kotu and Deshpande, 2014).

	Height (m)	Weight (kg)	HP	Attack	Defense	Speed	SP Attack	SP defense
Mean	1.27	69.75	69.40	80.09	74.30	68.47	72.68	71.96
Standard Deviation	1.39	129.22	25.76	32.37	30.83	29.75	32.65	27.54
Minimum	0.1	0.1	1.0	5.0	5.0	5.0	10.0	20.0
Median	1.0	28.5	66.0	76.0	70.0	65.0	65.0	70.0
Maximum	20.0	999.9	255.0	190.0	230.0	180.0	194.0	230.0

Table 1. Descriptive statistics.

1.2.3. Clustering methods

K-means clustering

Brief mechanism of k-means

K-means clustering is an unsupervised machine learning method and it can divide the data into clusters with similar items. Repeatedly assigning data point to cluster centroids and re-calculating centroid position before re-assignments, it can separate groups within data with the number of groups we want (Lantz, 2013).

Algorithm

A simple iterative clustering algorithm (Lloyd, 1957) was used which separates the dataset into clusters where each data point belongs to only one group (Awad & Khanna, 2015).

$$\mathbf{J} = \sum_{j=1}^{K} \sum_{n \in S_j} | \boldsymbol{x}_n - \boldsymbol{\mu}_j |^2 ,$$

where
x_n = vector representing the n^{th} data point
μ_j = geometric centroid of the data points in S_j

Parameters

Initial centroids were randomly generated. For each clustering run, 25 trials with different initial centroids were done and the best one was reported. Max iteration was defaulted as 10, meaning cluster assignment and moving of centroid were done for a maximum of 10 times.

Hierarchical clustering

Brief mechanism of Hierarchical clustering

Hierarchical clustering spread the basic clustering mission by request and the created clustering model is hierarchical. Points of intersection of a cluster hierarchy show clusters and their offspring are their sub-clusters (Cichosz, 2015).

Algorithm

$$h(x, l) = \begin{cases} d_0 & \text{if } \ l = 0 \\ h_{h(x,l-1)}(x) & \text{otherwise} \end{cases}$$

Hierarchical clustering algorithm divides a dataset into a sequence of nested partitions. The agglomerative algorithm was applied which works in a bottom-up manner and repeats merging clusters until a single cluster is emerged (Gan, 2011).

Parameters

The linkage method used is “ward.d2” based on Ward’s method, which considers the increase of sum of squares while merging cluster as the distance between clusters (Ward, 1963).

Euclidean distance

Both K-means and hierarchical clustering involve distance between data points and Euclidean distance was utilised. The Euclidean distance is normally defined to be smallest distance among the two points in Euclidean space (Liberti and Lavor, 2017). Cohen (2004) proposed the computation of distance in n-dimensional Euclidean distance and the distance formula is shown as below.

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

Methods to estimate suitable number of clusters

Elbow method

An “elbow” in the plot of average dispersion of data point to centroids describes the value of K that has improvement of dispersion declines the most, which is considered optimal number of cluster (Hackeling, 2014; Dangeti, 2017).

Silhouette method

The Silhouette method evaluates the quality of the clusters for each number of K. It measures how closely related an object to the own cluster against with the others. (Hwang, 2018).

Gap statistic

The gap statistic calculates a range of cluster values. The optimal number of clusters K is normally the highest gap between logW and E.logW in the result (Toomey, 2014).

1.2.4. Validation

Internal validation

Silhouette index

Pierson (2017) pointed out that it measures the average width of each point from all other points in a cluster. It also compares the average width of value to every point in every other cluster.

Dunn index

Cichosz (2015) mentioned that it measures the quality of a clustering model by relating the minimum separation to the maximum diameter displayed by clusters. Dunn index (D) can be calculated as follow:

$$D = \frac{min.\,separation}{max.\,diameter}$$

Visual inspections

PCA plots (for three or more attributes) or scatter plots (for two attributes) simulated distribution of data points and clusters. Boxplots illustrated the differences in attributes between clusters.

External validation

It was not applicable in current study since there is no "true" label in the data to justify the grouping of data points by the considered attributes.

1.3. Results

1.3.1. Estimation of number of clusters

Only the result from scenario A was illustrated as all models generated similar results.

Elbow method

The "elbow" in the within-cluster sum of square plot (Figure 1) lied on k = 3 although it was not very clearly observable.

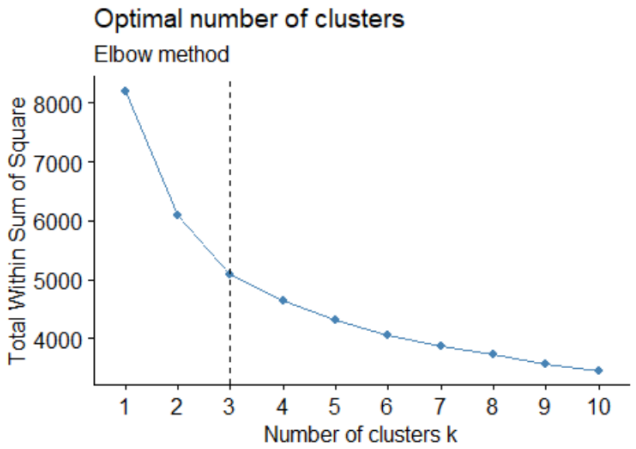


Figure 1. Within-cluster sum of square by number of clusters (k).

Silhouette method

The silhouette width plot suggested number of clusters was 3 (Figure 2).

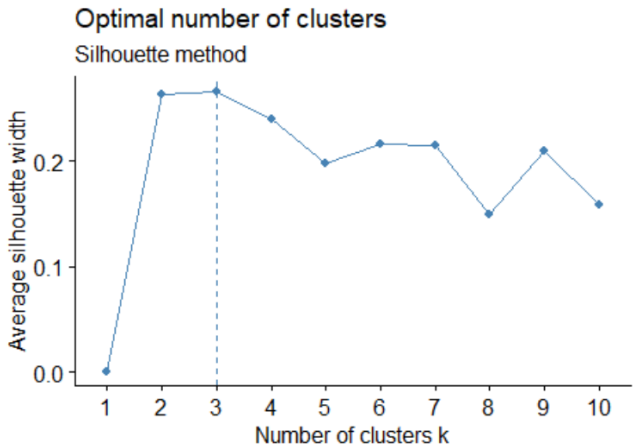


Figure 2. Average Silhouette width by number of clusters (k).

Gap statistic

Gap statistic suggested 5 as the optimal number of clusters (Figure 3). However, according to the 1-standard-error method suggested by Tibshirani et al. (2001), the optimal k number (k') should be the smallest k such that Gap(k) ≥ Gap(k+1) − sk+1, where s is standard error. In that case, the optimal number of clusters could be 2.

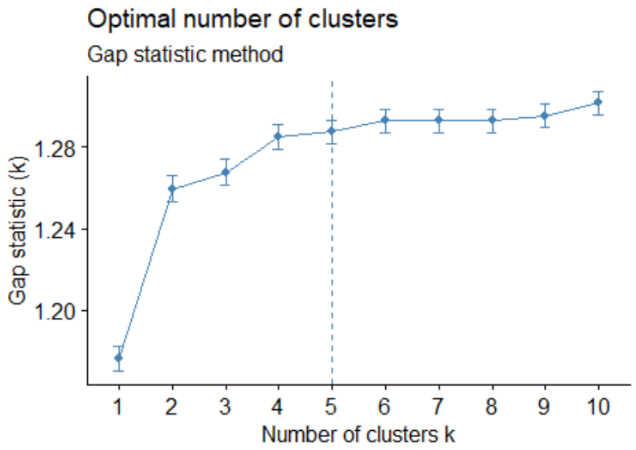


Figure 3. Gap statistic by number of cluster (k).

Decision

Since both elbow and silhouette methods were against k = 5 while k = 2 and k = 3 were supported, the experiments were carried out with the latter settings.

1.3.2. K-means clustering

```
# K-means clustering codes
kx_eclust <- eclust(pokemon_z[,c(1:8)], "kmeans", k=x, nstart=25, graph =
TRUE)
```

When k = 2, Pokémons were grouped into "stronger" and “weaker” clusters evenly in general (see Table 2 and Figure 4). The “stronger” cluster had all attributes higher (see Figure 5, boxplot for scenario A as example).

Scenarios		K-Means clustering					Hierarchical clustering				
		k=2		k=3			2 clusters		3 clusters		
		cluster 1	cluster 2	cluster 1	cluster 2	cluster 3	cluster 1	cluster 2	cluster 1	cluster 2	cluster 3
A. 8 attributes (HP, attack, defense, SP attack, SP defense, speed, height, weight)	**Size**	**484**	**543**	**61**	**422**	**544**	**383**	**644**	**383**	**616**	**28**
	Mean Attack	59.57	98.37	123.92	57.15	92.96	55.43	94.75	55.43	93.51	122.07
	Mean Defense	57.01	89.72	117.74	53.56	85.53	52.93	87.02	52.93	85.55	119.39
	Mean Height	0.64	1.83	5.01	0.61	1.36	0.60	1.67	0.60	1.44	6.89
	Mean HP	53.61	83.47	106.08	51.50	79.17	50.71	80.51	50.71	79.27	107.75
	Mean SP attack	52.84	90.36	104.08	50.94	86.03	49.18	86.66	49.18	85.70	107.68
	Mean SP defense	54.04	87.93	98.57	51.43	84.90	50.56	84.68	50.56	84.18	95.89
	Mean Speed	54.96	80.52	76.26	53.48	79.23	50.97	78.88	50.97	79.17	72.46
	Mean Weight	18.74	115.22	467.94	16.72	66.24	15.51	102.01	15.51	78.09	628.29
B. 6 attributes (HP, attack, defense, SP attack, SP defense, speed)	**Size**	**449**	**578**	**315**	**303**	**409**	**305**	**722**	**305**	**577**	**145**
	Mean Attack	58.22	97.07	99.19	92.14	56.44	51.83	92.02	51.83	86.05	115.79
	Mean Defense	55.07	89.25	76.66	100.94	52.76	48.03	85.41	48.03	83.82	91.72
	Mean HP	52.72	82.35	78.50	84.01	51.56	48.47	78.24	48.47	75.89	87.59
	Mean SP attack	51.47	89.16	100.22	74.14	50.40	47.66	83.25	47.66	73.60	121.68
	Mean SP defense	52.62	86.98	84.23	87.67	50.87	48.25	81.98	48.25	78.24	96.83
	Mean Speed	53.64	79.99	101.18	54.91	53.33	50.70	75.98	50.70	69.71	100.94
C. 4 attributes (Hp, attack, defense, speed)	**Size**	**546**	**481**	**324**	**279**	**424**	**441**	**586**	**441**	**319**	**267**
	Mean Attack	100.80	56.57	100.32	94.60	55.07	58.96	95.99	58.96	86.08	107.82
	Mean Defense	88.99	57.63	76.50	102.72	53.93	68.08	78.99	68.08	64.95	95.76
	Mean HP	83.90	52.93	77.08	87.36	51.71	52.48	82.13	52.48	69.60	97.09
	Mean Speed	81.02	54.23	101.41	53.63	53.08	47.22	84.47	47.22	94.46	72.53
D. 3 attributes (Hp, attack, defense)	**Size**	**484**	**543**	**174**	**342**	**511**	**567**	**460**	**567**	**382**	**78**
	Mean Attack	103.51	59.20	91.23	106.49	58.62	60.22	104.58	60.22	107.51	90.19
	Mean Defense	93.77	56.95	120.95	79.94	54.65	59.09	93.06	59.09	84.48	135.09
	Mean HP	85.80	54.78	67.37	92.51	54.62	55.06	87.07	55.06	91.50	65.36
E. Only height and weight	**Size**	**969**	**58**	**882**	**24**	**121**	**994**	**33**	**887**	**107**	**33**
	Mean Height	1.04	5.09	0.92	6.85	2.71	1.09	6.74	0.93	2.46	6.74
	Mean Weight	44.90	485.05	31.91	703.08	220.01	53.14	570.15	32.78	221.91	570.15
F. Only SP attack and SP defense	**Size**	**451**	**576**	**398**	**222**	**407**	**543**	**484**	**543**	**334**	**150**
	Mean SP attack	98.79	52.24	45.44	114.71	76.39	48.77	99.50	48.77	92.64	114.79
	Mean SP defense	94.21	54.54	48.15	105.62	76.88	57.42	88.27	57.42	77.13	113.09

Table 2. Result cluster sizes and means of attributes for each cluster by K-means and hierarchical clustering in all scenarios.

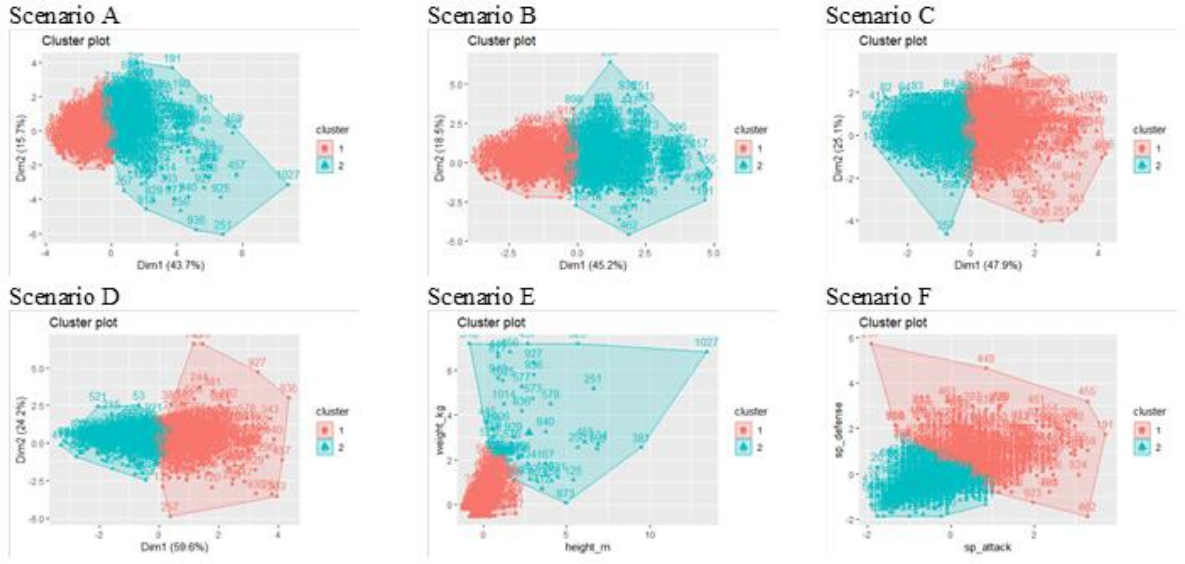


Figure 4. PCA/Scatter plots for K-means (k=2) clustering in each scenario.

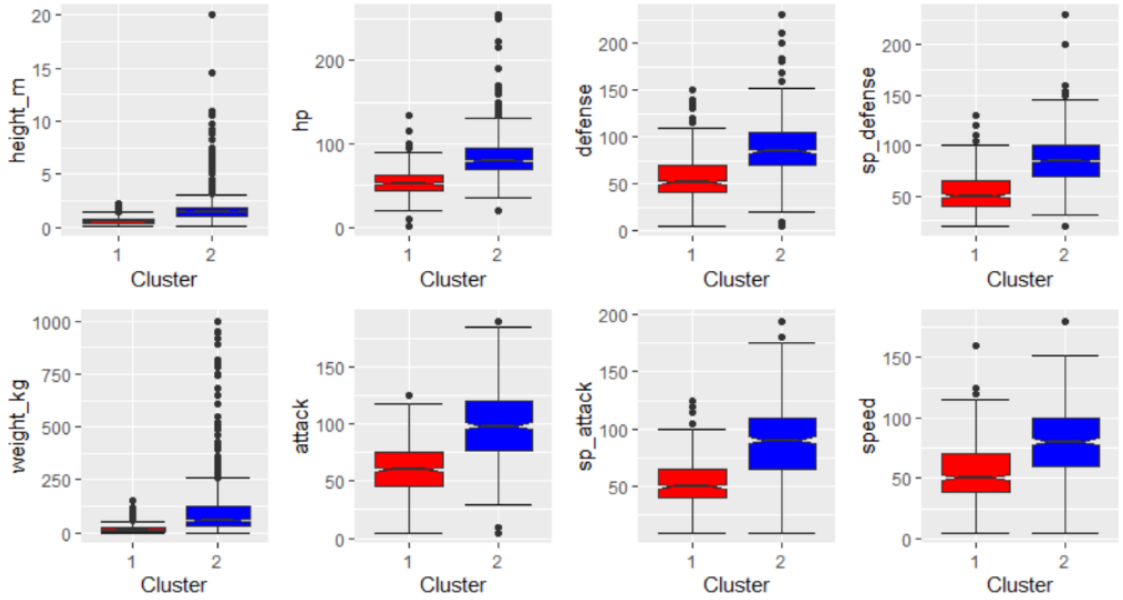


Figure 5. Boxplots for each attribute in K-means clustering (k=2) in scenario A. The notch of each box indicated the confidence interval around the median.

When k=3, the result for scenario A was influenced by height and weight, forming a small cluster of "top-tier" Pokémons (see Table 2 and Figure 6) with all attributes higher, except speed (see Figure 7).

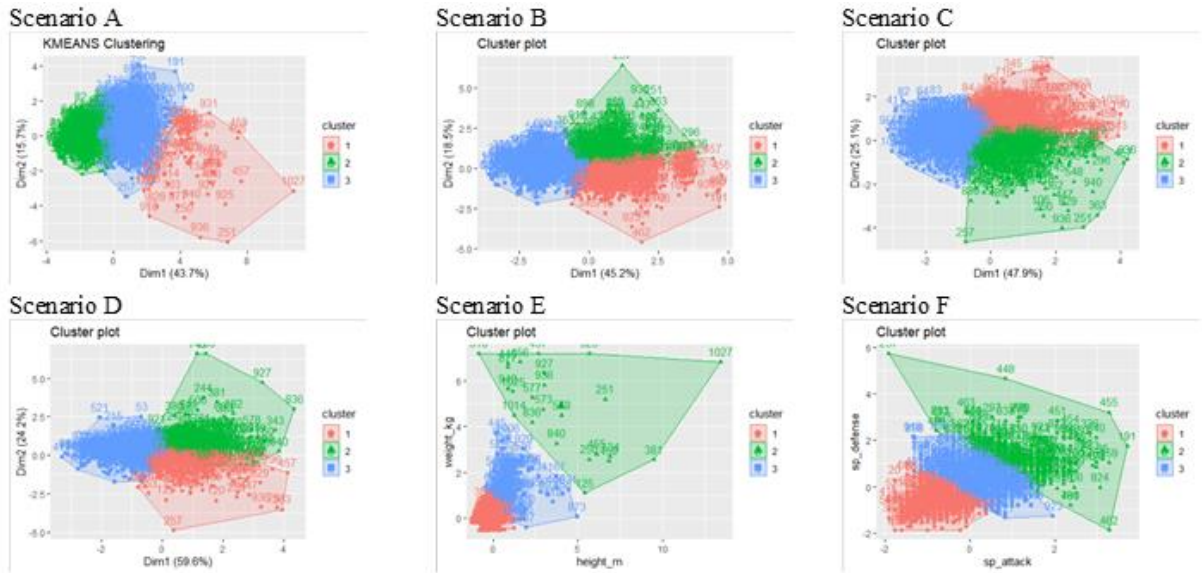


Figure 6. PCA plots for K-means (k=3) clustering in each scenario.

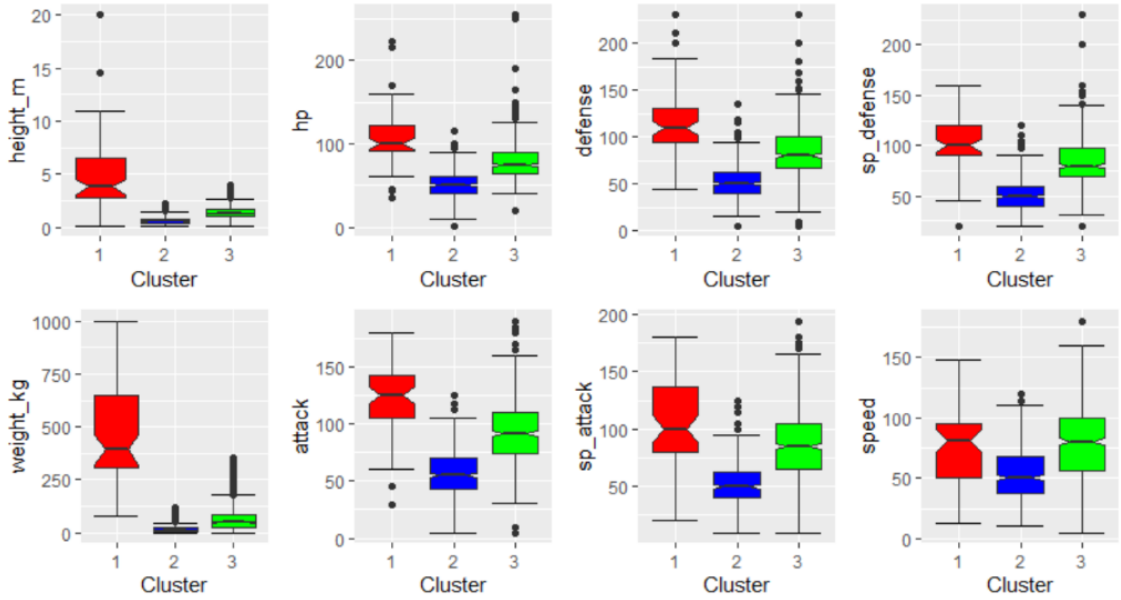


Figure 7. Boxplots for each attribute in K-means clustering (k=3) in scenario A.

For scenarios B -D, clusters with higher speed, attack and SP attack contrasted with the higher defense and HP clusters (see Figure 8).

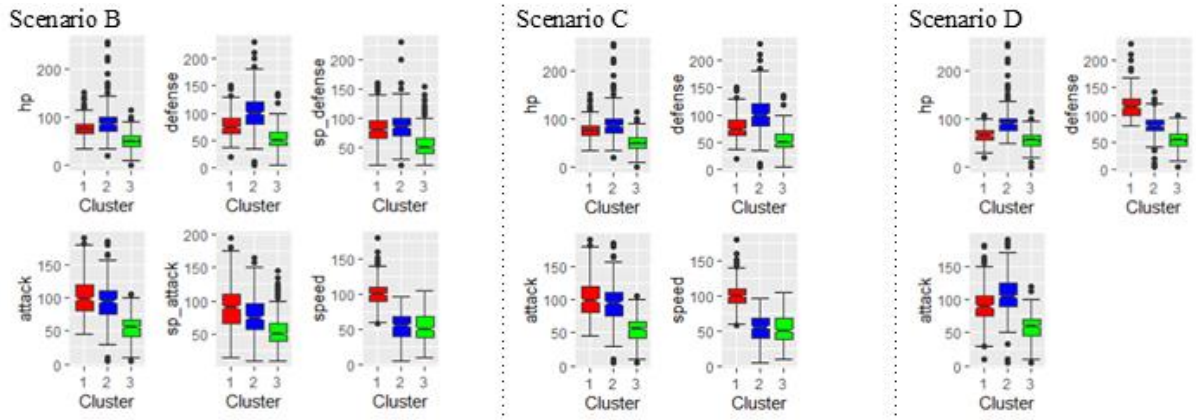


Figure 8. Boxplots for each attribute in K-means clustering (k=3) in scenarios B-D.

1.3.3. Hierarchical clustering

```
# Hierarchical clustering codes
hcx_all <- eclust(pokemon_z, "hclust", k=x, hc_method="ward.D2", graph =
TRUE)
```

Hierarchical clustering yielded similar results to K-means clustering, splitting the data into “stronger” and “weaker” categories and “top-tier” Pokémons were divided from the “stronger” category in scenarios A and B (see Table 2 and dendrograms in Figure 9). In scenarios C and D, clusters were determined by speed, attack, defense, and HP (see Figure 10).

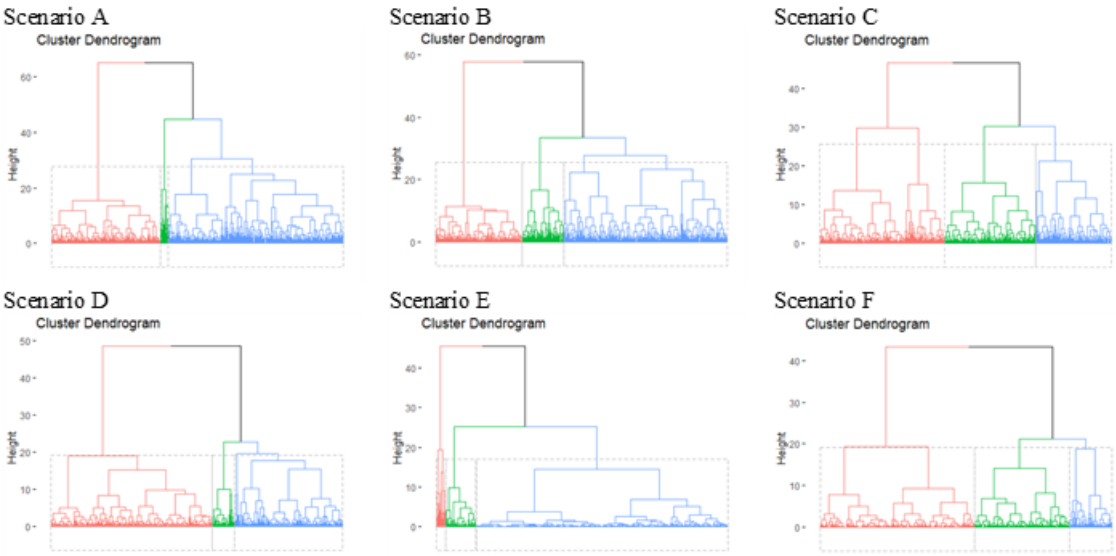


Figure 9. Dendrograms for hierarchical clustering in all scenarios. Cuts for 3 clusters were illustrated with dotted rectangles.

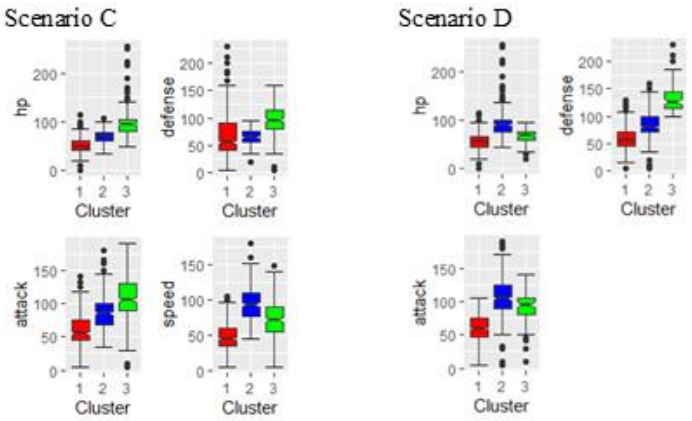


Figure 10. Boxplots for hierarchical clustering (3 clusters) in scenarios C & D.

1.3.4. Validation

The average silhouette widths for most cases were about 0.3 which were only acceptable (see Table 3). Negative silhouette widths indicated mis-assignments of some data points (see Figure 11).

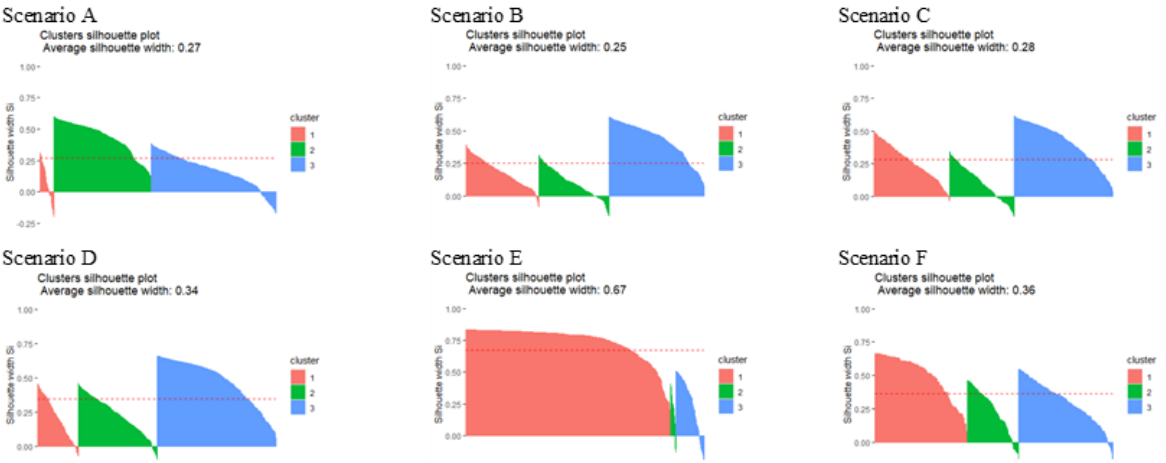


Figure 11. Silhouette plots for K-means clustering (k=3) for each scenario.

Noticeably low Dunn indices of below 0.04 were found in all models. it might relate to large diameter of one or more clusters.

Visual inspection of plots could not see clusters heavily overlapped, but they were very closely located, displaying the results were unexceptional.

Among all models, the 3-cluster settings in scenario A and B showed better stability.

		K-means clustering		Hierarchical clustering	
Scenarios	**Indices**	**2 clusters**	**3 clusters**	**2 clusters**	**3 clusters**
A. 8 attributes (HP, attack, defense, SP attack, SP defense, speed, height, weight)	Average silhouette width	0.2628	0.2676	0.2391	0.2584
	Dunn index	0.0262	0.0272	0.0317	0.0378
B. 6 attributes (HP, attack, defense, SP attack, SP defense, speed)	Average silhouette width	0.2827	0.2535	0.2438	0.197
	Dunn index	0.0252	0.0274	0.0357	0.0388
C. 4 attributes (Hp, attack, defense, speed)	Average silhouette width	0.295	0.2816	0.2543	0.2094
	Dunn index	0.0178	0.0193	0.0268	0.0268
D. 3 attributes (Hp, attack, defense)	Average silhouette width	0.3645	0.3449	0.3585	0.3326
	Dunn index	0.0067	0.0081	0.0118	0.0136
E. Only height and weight	Average silhouette width	0.8125	0.6697	0.8449	0.6519
	Dunn index	0.0078	0.0051	0.0358	0.005
F. Only SP attack and SP defense	Average silhouette width	0.4471	0.3625	0.3985	0.3292
	Dunn index	0.0033	0.0051	0.01	0.012

Table 3. Average silhouette widths and Dunn indices for each scenario with K-means and hierarchical clustering.

1.3.4. Comparison with existing categories

“Weaker” cluster contained mostly Normal Pokémons. Other status was not particularly defined in any of the clusters (see Table 4).

The distribution of Pokémon types (see example in Figure 12) indicated no relationship between types and clusters.

		K-Means clustering					Hierarchical clustering				
		k=2		k=3			2 clusters		3 clusters		
Scenarios		cluster 1	cluster 2	cluster 1	cluster 2	cluster 3	cluster 1	cluster 2	cluster 1	cluster 2	cluster 3
A. 8 attributes (HP, attack, defense, SP attack, SP defense, speed, height, weight)	**Size**	**484**	**543**	**61**	**422**	**544**	**383**	**644**	**383**	**616**	**28**
	Normal	481 (99.4%)	434 (79.9%)	22 (36.1%)	419 (99.3%)	474 (87.1%)	381 (99.5%)	534 (82.9%)	381 (99.5%)	525 (85.2%)	9 (32.1%)
	Legendary	1 (0.2%)	37 (6.8%)	30 (49.2%)	1 (0.2%)	7 (1.3%)	1 (0.3%)	37 (5.7%)	1 (0.3%)	23 (3.7%)	14 (50.0%)
	Sub Legendary	1 (0.2%)	44 (8.1%)	6 (9.8%)	1 (0.2%)	38 (7.0%)	0 (0.0%)	45 (7.0%)	0 (0.0%)	42 (6.8%)	3 (10.7%)
	Mythical	1 (0.2%)	28 (5.2%)	3 (4.9%)	1 (0.2%)	25 (4.6%)	1 (0.3%)	28 (4.3%)	1 (0.3%)	26 (4.2%)	2 (7.1%)
B. 6 attributes (HP, attack, defense, SP attack, SP defense, speed)	**Size**	**449**	**578**	**315**	**303**	**409**	**305**	**722**	**305**	**577**	**145**
	Normal	445 (99.1%)	470 (81.3%)	231 (73.3%)	278 (91.7%)	406 (99.3%)	303 (99.3%)	612 (84.8%)	303 (99.3%)	548 (95.0%)	386 (94.8%)
	Legendary	2 (0.4%)	36 (6.2%)	31 (9.8%)	6 (2.0%)	1 (0.2%)	1 (0.3%)	37 (5.1%)	1 (0.3%)	3 (0.5%)	4 (1.0%)
	Sub Legendary	1 (0.2%)	44 (7.6%)	30 (9.5%)	14 (4.6%)	1 (0.2%)	0 (0.0%)	45 (6.2%)	0 (0.0%)	18 (3.1%)	11 (2.7%)
	Mythical	1 (0.2%)	28 (4.8%)	23 (7.3%)	5 (1.7%)	1 (0.2%)	1 (0.3%)	28 (3.9%)	1 (0.3%)	8 (1.4%)	6 (1.5%)
C. 4 attributes (HP, attack, defense, speed)	**Size**	**546**	**481**	**324**	**279**	**424**	**441**	**586**	**441**	**319**	**267**
	Normal	439 (80.4%)	476 (99.0%)	249 (76.9%)	246 (88.2%)	420 (99.1%)	432 (98.0%)	483 (82.4%)	432 (98.0%)	286 (89.7%)	197 (73.8%)
	Legendary	36 (6.6%)	2 (0.4%)	24 (7.4%)	12 (4.3%)	2 (0.5%)	2 (0.5%)	36 (6.1%)	2 (0.5%)	2 (0.6%)	34 (12.7%)
	Sub Legendary	43 (7.9%)	2 (0.4%)	28 (8.6%)	16 (5.7%)	1 (0.2%)	4 (0.9%)	41 (7.0%)	4 (0.9%)	17 (5.3%)	24 (9.0%)
	Mythical	28 (5.1%)	1 (0.2%)	23 (7.1%)	5 (1.8%)	1 (0.2%)	3 (0.7%)	26 (4.4%)	3 (0.7%)	14 (4.4%)	12 (4.5%)
D. 3 attributes (HP, attack, defense)	**Size**	**484**	**543**	**174**	**342**	**511**	**567**	**460**	**567**	**382**	**78**
	Normal	379 (78.3%)	536 (98.7%)	149 (85.6%)	260 (76.0%)	506 (99.0%)	558 (98.4%)	357 (77.6%)	558 (98.4%)	291 (76.2%)	66 (84.6%)
	Legendary	35 (7.2%)	3 (0.6%)	5 (2.9%)	31 (9.1%)	2 (0.4%)	2 (0.4%)	36 (7.8%)	2 (0.4%)	35 (9.2%)	1 (1.3%)
	Sub Legendary	42 (8.7%)	3 (0.6%)	13 (7.5%)	30 (8.8%)	2 (0.4%)	5 (0.9%)	40 (8.7%)	5 (0.9%)	33 (8.6%)	7 (9.0%)
	Mythical	28 (5.8%)	1 (0.2%)	7 (4.0%)	21 (6.1%)	1 (0.2%)	2 (0.4%)	27 (5.9%)	2 (0.4%)	23 (6.0%)	4 (5.1%)
E. Only height and weight	**Size**	**969**	**58**	**882**	**24**	**121**	**994**	**33**	**887**	**107**	**33**
	Normal	892 (92.1%)	23 (39.7%)	829 (94.0%)	8 (33.3%)	78 (64.5%)	903 (90.8%)	12 (36.4%)	832 (93.8%)	71 (66.4%)	12 (36.4%)
	Legendary	12 (1.2%)	26 (44.8%)	3 (0.3%)	11 (45.8%)	24 (19.8%)	22 (2.2%)	16 (48.5%)	3 (0.3%)	19 (17.8%)	16 (48.5%)
	Sub Legendary	39 (4.0%)	6 (10.3%)	25 (2.8%)	3 (12.5%)	17 (14.0%)	42 (4.2%)	3 (9.1%)	27 (3.0%)	15 (14.0%)	3 (9.1%)
	Mythical	26 (2.7%)	3 (5.2%)	25 (2.8%)	2 (8.3%)	2 (1.7%)	27 (2.7%)	2 (6.1%)	25 (2.8%)	2 (1.9%)	2 (6.1%)
F. Only SP attack and SP defense	**Size**	**451**	**576**	**398**	**222**	**407**	**543**	**484**	**543**	**334**	**150**
	Normal	347 (76.9%)	568 (98.6%)	394 (99.0%)	135 (60.8%)	386 (94.8%)	535 (98.5%)	380 (78.5%)	535 (98.5%)	294 (88.0%)	86 (57.3%)
	Legendary	36 (8.0%)	2 (0.3%)	1 (0.3%)	33 (14.9%)	4 (1.0%)	2 (0.4%)	36 (7.4%)	2 (0.4%)	7 (2.1%)	29 (19.3%)
	Sub Legendary	41 (9.1%)	4 (0.7%)	2 (0.5%)	32 (14.4%)	11 (2.7%)	5 (0.9%)	40 (8.3%)	5 (0.9%)	18 (5.4%)	22 (14.7%)
	Mythical	27 (6.0%)	2 (0.3%)	1 (0.3%)	22 (9.9%)	6 (1.5%)	1 (0.2%)	28 (5.8%)	1 (0.2%)	15 (4.5%)	13 (8.7%)

Table 4. Cluster distribution with Pokémon status (Normal, Legendary, Sub-legendary and Mythical). Percentage in blankets referred to the proportion to cluster size.

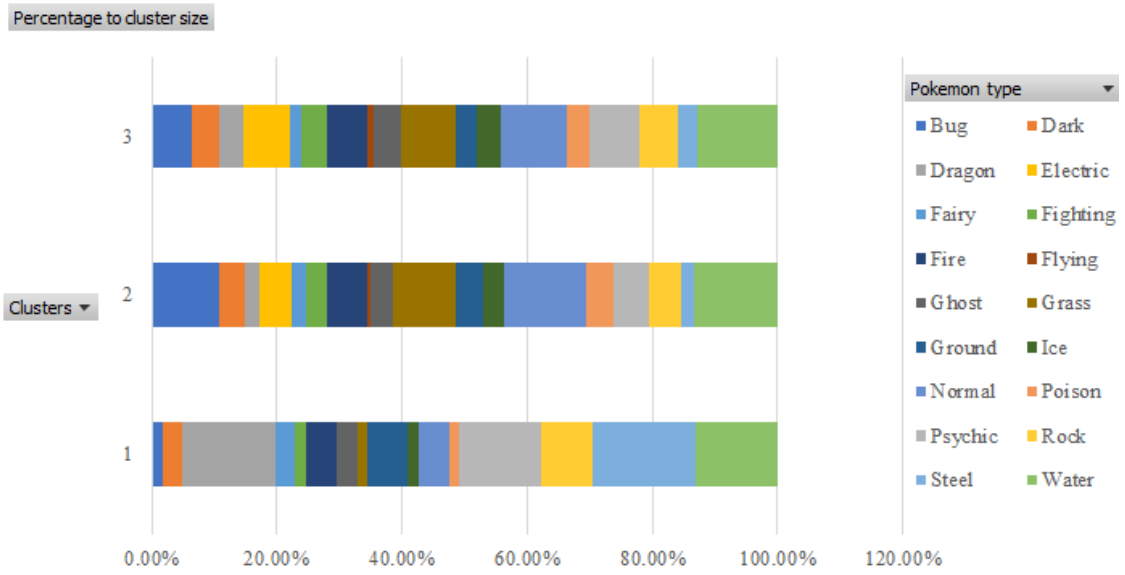


Figure 12. Distribution of Pokémon types in each cluster in K-means clustering (k=3) for scenario A.

1.4. Brief discussion

The 3-cluster K-means clustering model with 6 attributes was considered the best model in current study, due to its decent stability and more meaningful grouping of Pokémons compared other models.

The 3 clusters from the final model divided Pokémons into one “weaker” group and two “stronger” groups: “Speed-Attack”, which had higher speed and attack, and “Hard-Defense”, which had higher HP and defense. “Weaker” Pokémons were easier to catch (Figure 13). Players can pick the Pokémons from these groups based on their progresses and strategies.

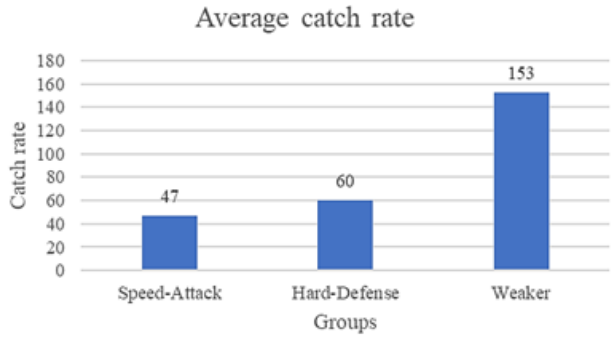


Figure 13. Average catch rate for each group of Pokémons.

The quality of this model was not excellent as indicated by validation indices and overlapping of cluster boundaries. Few marginal cases might be assigned inappropriately.

To improve quality of results, alternative clustering methods may be considered for cross-validation. K-medoids, or Partitioning Around Medoids (PAM) (Kaufman & Rousseeuw, 1990) is less sensitive to outliers than K-means despite its low efficiency (Park & Jun, 2009). It may be useful to consolidate K-means results. Fuzzy clustering, which allows data point to have a degree of membership in different clusters, provides flexibility on assigning bridges or undetermined data (Ruspini, 1969). It may produce refined results for current data which contains substantial marginal cases.

1.4. Study 1 conclusion

Exploratory clustering was performed on Pokémon data to categorise Pokémons based on their basic stats. After testing models with various settings and methods, a 3-cluster model based on 6 attributes was proposed to define three major groups of Pokémons. Implications, quality of results and alternative methods were discussed.

2. Study 2: Can legendary Pokémon be identified using classification systems, specifically decision tree and KNN methods (James & Daniel)

2.1.1. Introduction

The aim of this research was to evaluate a decision tree's use in identifying and classifying legendary Pokémon in the chosen dataset.

2.1.2. Laptop Configuration

Operating System	Windows 10 Pro 64-bit
Processor	Intel Core i7-8650U @ 1.9 GHz
RAM	16GB

Figure 2.1.1: Laptop configuration

2.1.3. Dataset Exploration

```
df %>% ggplot() +
  geom_bar(aes( x = status, fill = status))+
  theme_classic()+
  theme(legend.title = element_blank())+
  labs(title = "Count of Pokemon Status")
```

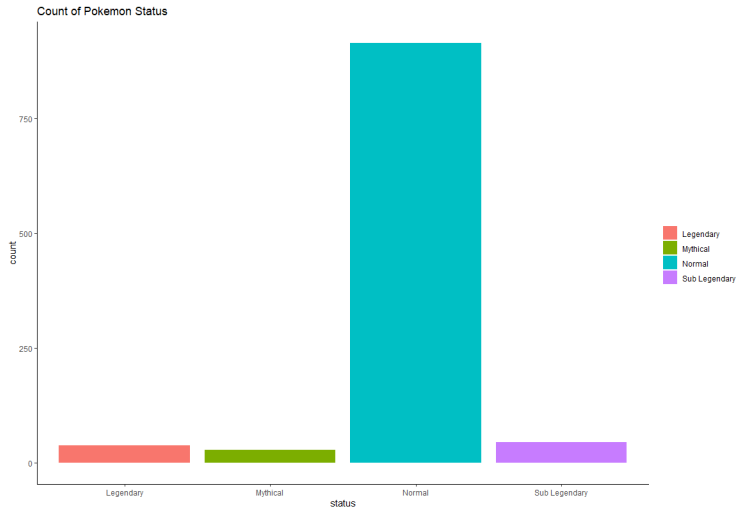


Figure 2.1.2: Count of Pokémon by Status

```
pokemon_stats <- df %>% select(status, height_m, weight_kg, hp, attack,
defense, sp_attack, sp_defense, speed)
pokemon_stats_legendary <- pokemon_stats %>% filter(status ==
"Legendary")
stats_leg_long <- pokemon_stats_legendary %>% select(-status, -
height_m, -weight_kg) %>%
  gather(attribute, value)

stats_leg_long %>% ggplot()+
  geom_boxplot(aes(x=attribute, y=value, fill= attribute), show.legend
= FALSE)+
  theme_classic()+
  labs(title = "Legendary Pokemon Stats")
```

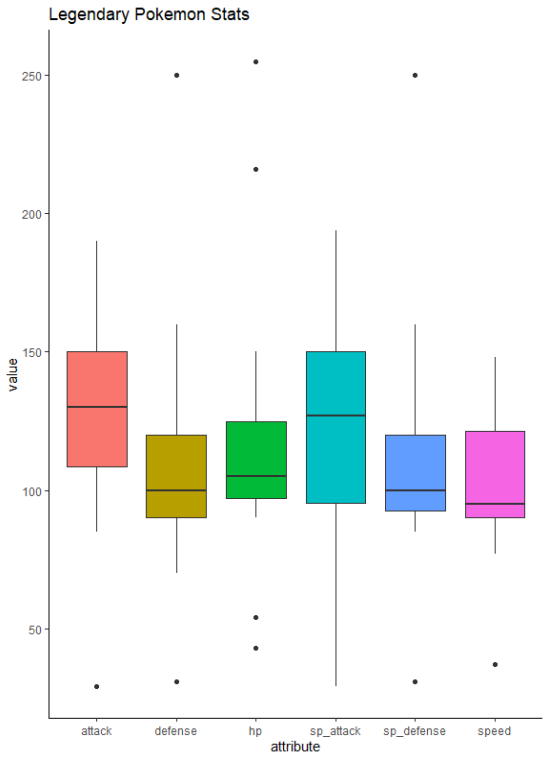


Figure 2.1.3: Distribution of Basic Stats by Legendary Pokémon

```
pokemon_stats_nonlegendary <- pokemon_stats %>% filter(status != 
"Legendary")
stats_nonleg_long <- pokemon_stats_nonlegendary %>% select(-status, -
height_m, -weight_kg) %>%
  gather(attribute, value)

stats_nonleg_long %>% ggplot()+
  geom_boxplot(aes(x=attribute, y=value, fill= attribute), show.legend
= FALSE)+
  theme_classic()+
  labs(title = "Non-Legendary Pokemon Stats")
```

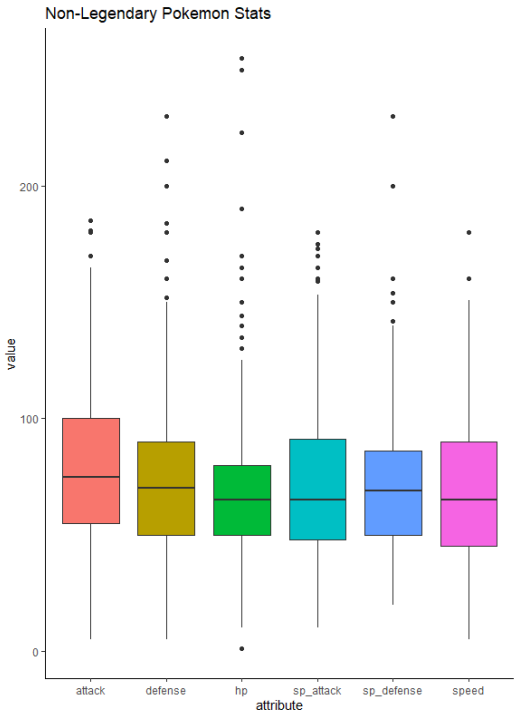


Figure 2.1.4: Distribution of Basic Stats by Non-Legendary Pokémon

```
pokemon_wh_leg <- pokemon_stats_legendary %>% select(status, height_m,
weight_kg)
pokemon_wh_nonleg <- pokemon_stats_nonlegendary %>% select(status,
height_m, weight_kg)

ggplot() +
  geom_jitter(data = pokemon_wh_nonleg, aes(x= weight_kg, y= height_m,
colour = status), alpha = 0.5, height = 0.04)+
  geom_point(data = pokemon_wh_leg, aes(x= weight_kg, y= height_m,
colour = status), size = 3.5)+
  scale_y_log10()+
  theme_classic()+
  theme(legend.title = element_blank())+
  labs(title = "Pokemon Height & Weight by Status")+
  xlab(label = "Weight (kg)")+
  ylab(label = "Height (m)")
```

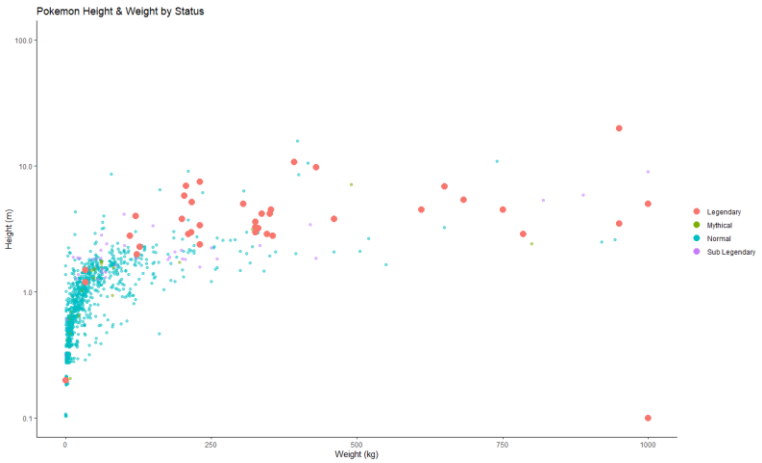


Figure 2.1.5: Correlation Between Height & Weight by Status

2.1.4. Data Preparation

```
df <- df %>% mutate(legendary = case_when(
  status == "Legendary" ~ 1,
  TRUE ~ 0))

df_stats <- df %>% select(legendary, generation, type_1, type_2,
height_m, weight_kg, total_points, hp, attack, defense,
sp_attack, sp_defense, speed, catch_rate, base_friendship,
base_experience, growth_rate, percentage_male, egg_cycles)

df_stats$legendary <- as.factor(df_stats$legendary)
df_stats$generation <- as.factor(df_stats$generation)
df_stats$type_1 <- as.factor(df_stats$type_1)
df_stats$type_2 <- as.factor(df_stats$type_2)
df_stats$growth_rate <- as.factor(df_stats$growth_rate)
```

Pokemon with a ‘Legendary’ class were given an identifier of 1 and all others 0. Redundent columns were removed from the dataset leaving only the needed attribute columns required for classification and caterigorical attributes converted to factors.

```
set.seed(42)
split_index <- sample(1:nrow(df_stats), round(nrow(df_stats) * 0.8))
train <- df_stats[split_index, ]
test <- df_stats[-split_index, ]
```

The set.seed() function was inititated prior to any data splitting for reproducibility of any findings, the dataset was then split into train & test sets using a 70/30 ratio. The reasoning behind data splitting is that we want an algorithm that performs well on both known and unknown data.

2.1.5. Model

The most common implementation of Decision Trees in R is through the use of a package known as ‘rpart’, the latter making use of the CART or Classification & Regression Trees methodology. Each region of the tree is then continuously divided into smaller sub-groups based on splitting rules until an absolute outcome is reached. As CART is binary recursive in nature i.e. a hierarchy of logical outcomes, no normalisation or standardisation of the data is required (Breiman, 2017).

```
df_tree_all <- rpart(formula = legendary ~ .,
                     data = train,
                     method = "class",
                     control = list(cp = 0))
```

By default rpart uses Gini index as a measure of error, the latter is an attempt to minimise node impurity i.e. consisting mostly of observations from a single class (Boehmke et al, 2020).

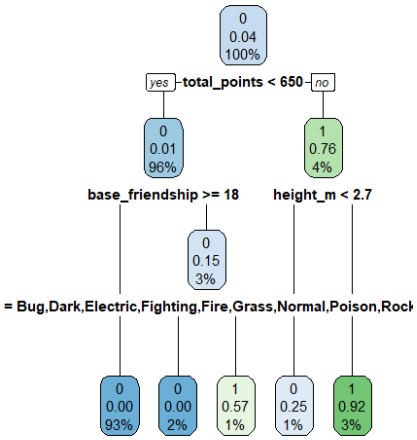


Figure 2.1.6: Non-Pruned Decision Tree

2.1.6 Model Tuning

The tree is optimised by pruning, the process of growing a decision tree to its largest depth and scaling it back to an optimal point. The latter is achieved by penalising the measure used with a complexity parameter (Boehmke et al, 2020).

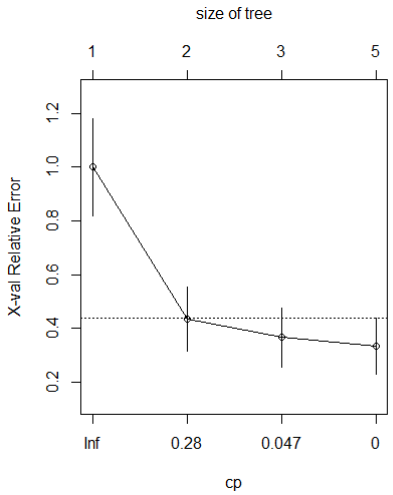


Figure 2.1.7: Complexity Paramter Plot

Breiman (2017) suggests the use of the 1-SE rule, selecting a complexity measure or tree size within 1 standard error of the minimum cross validation error. The latter is calculated as 0.33333 + 0.10477 = 0.4381 which would result in a less complex tree.

```
df_tree_1SE <- prune(df_tree_all, cp = 0.133333)
```

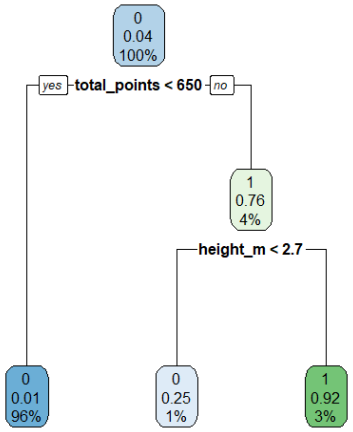


Figure 2.1.8: Pruned Decision Tree

Decision trees automatically select attributes based on their importance in error reduction. Both 'total_points' and 'height_m' are used in the pruned tree as above and rank highest as shown in figure below.

```
vip(df_tree_all, geom = "point")+
  theme_classic()+
  labs(title = "Classification Attributes by Importance")+
  xlab(label = "Attributes")+
  ylab(label = "Importance")
```

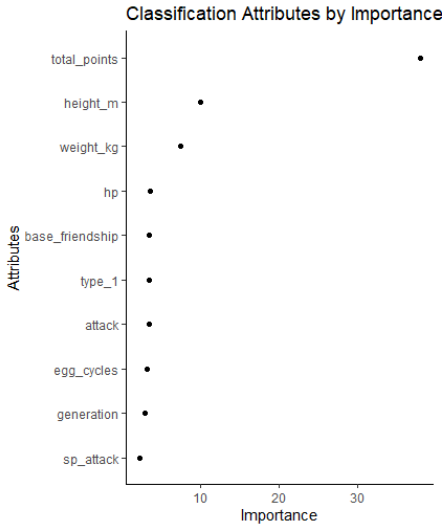


Figure 2.1.9: Plot of VIP Attributes, Ranked by Importance of Error Reduction

2.1.7 Model Evaluation

Classification models are often evaluated using a confusion matrix, a matrix comparing actual class instances to predicted instances. When an instance of a class is predicted correctly it is referred to as a 'true positive', an instance that is predicted incorrectly is referred to as a 'false positive'. The model has a high accuracy of 98.5% meaning it is often correct in classifying instances overall. Of those predicted as legendary 77.7% were actual instances (precision) and 87.5% of all legendary Pokémon were correctly identified (recall) (Boehmke et al, 2020).

Prediction	Reference	
	0	1
0	*196*	*2*
1	*1*	*7*

Accuracy :	*0.9854*
Precision :	*0.77778*
Recall :	*0.87500*
Specificity :	*0.99492*

Figure 2.1.10: Confusion Matrix Values

An additional measure of a model's accuracy can be analysed using an ROC curve plotted against the true positive rate and false positive rate. The true positive rate is the percentage of instances accurately predicted and the false positive rate is the percentage of instances incorrectly predicted, the greater the area under the curve the greater the model's accuracy. Figure 2.1.11 below correlates well with the results from the confusion matrix above (Irizarry, 2019).

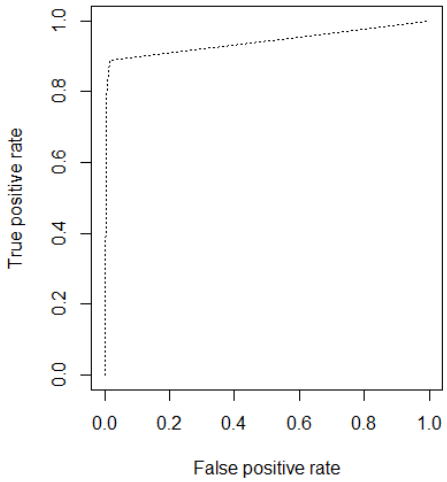


Figure 2.1.11: ROC Plot

2.1.8 Conclusion

Considering the class imbalance presented in the dataset the model appears to be fairly accurate in predicting legendary Pokemon based on their attributes. Model precision may potentially be further increased by experimenting with chosen complexity parameter values for pruning or by up-sampling the dataset due to the previously mentioned class imbalance.

KNN

2.2. Objective

This element of the ICA used the data from the Pokémon dataset to determine if the 'Legendary' status of Pokémon can be predicted using a range of stat categories utilising the K-Nearest-Neighbour method. It is believed that higher accuracy can be found by combining different stats and negating others. This analysis questions whether Legendary status can be determined using physical attributes such as height and weight, ordinary battle statistics like HP, attack, defence and speed, special attack and defence.

2.2.1. K Nearest Neighbour

K Nearest Neighbour is an algorithm used to classify a new data point by applying similarity measures against existing data points (Subramamanian, D., 2020), for example, identifying the stats that denote a Legendary status and comparing it to a new data point

The K in the algorithm is a parameter that identifies the number of 'neighbours', based on Euclidian distance, to compare the new data point to. A smaller value for K means that noise will have a higher influence on the result, and a large value make it computationally expensive.

Data scientists often determine K by finding the square root of the total number of data points in the dataset ((Zhang et al., 2018).

2.2.2. Analyses & Data Processing

The following six attribute combinations were analysed:

	Analysis 1	Analysis 2	Analysis 3	Analysis 4	Analysis 5	Analysis 6
Generation	x	x				x
Height	x		x			x
Weight			x			x
Total Points	x	x				x
HP	x	x				x
Attack	x	x			x	x
Defence	x	x			x	x
Special Attack	x	x		x		x
Special Defence	x	x		x		x
Speed	x				x	x

Table 2.2.1. Analyses based on attributes

Identifying NaN values in the dataset and negating them from the analysis:

```
pokeData.isnull().sum()
pokeData = pokeData.dropna()
pokeData.isnull().sum()
```

To obtain insight into the relevance of various stats in relation to Legendary status, a heatmap was developed.

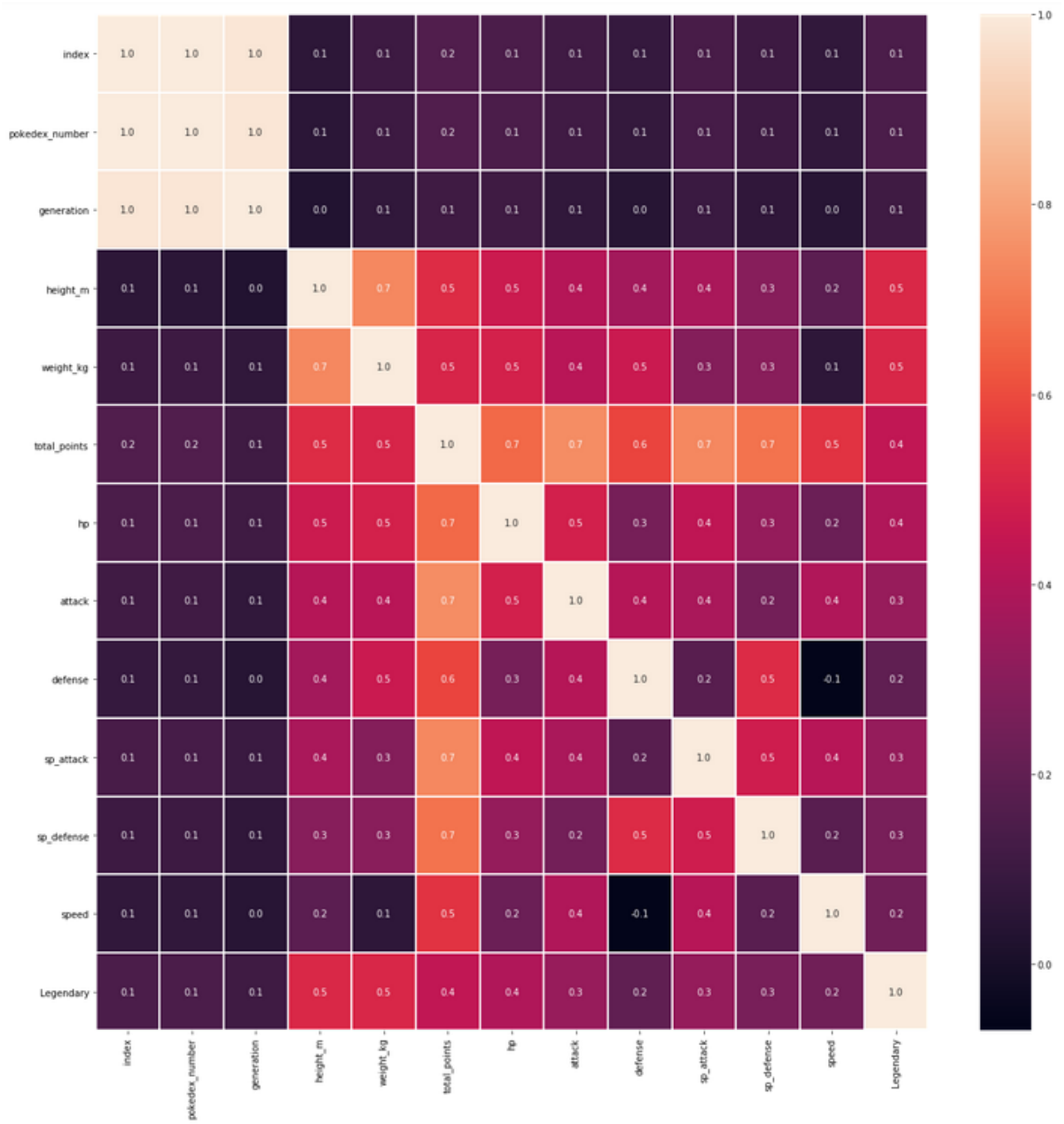


Figure 2.2.1. Attribute Correlation Heatmap

This visualisation shows that physical attributes like height and weight are more likely to indicate Legendary status.

It was next necessary to negate attributes from the dataframes to match the criteria in the six analyses. As the columns containing index, name, species, type and pokedex data were not required for the analysis, these were removed. An example of the code used to achieve this in Analysis 1 is provided below.

```
pokeData_knn= pokeData.copy()
pokeData_knn.drop(['name','type_1', 'type_2', 'status', 'species',
'weight_kg', 'pokedex_number', 'index'],axis=1, inplace=True)
```

	height_m	total_points	hp	attack	defense	sp_attack	sp_defense	speed	Legendary
0	0.7	318	45	49	49	65	65	45	False
1	1.0	405	60	62	63	80	80	60	False
2	2.0	525	80	82	83	100	100	80	False
3	2.4	625	80	100	123	122	120	80	False
6	1.7	534	78	84	78	109	85	100	False

2.2.4 Test-Train Split & K value

To split the data into test and training sets, test_train_split was imported from the python sklearn.model_selection library. Legendary status was set as the predicted attribute, along with a test size of 30% of the dataset. Initial analysis used K = 5 on each analysis, corrected for a second test after Error Rate vs K was determined.

```
pokeData_knn['Legendary'] = pokeData_knn['Legendary'].astype(int)
scaler = StandardScaler()
scaler.fit(pokeData_knn.drop('Legendary', axis = 1))
scaled_pokeData =  scaler.transform(pokeData_knn.drop('Legendary', axis
= 1))
scaled = pd.DataFrame(scaled_pokeData, columns =
pokeData_knn.columns[:-1])
X = scaled
Y = pokeData_knn['Legendary']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
.3, random_state = 75)
Knn = KNeighborsClassifier(K = 5)
knn.fit(X_train, y_train)
predictions = knn.predict(X_test)
```

```
accuracy = 0.988, precision = 0.988, recall = 0.988, f1 = 0.988
```

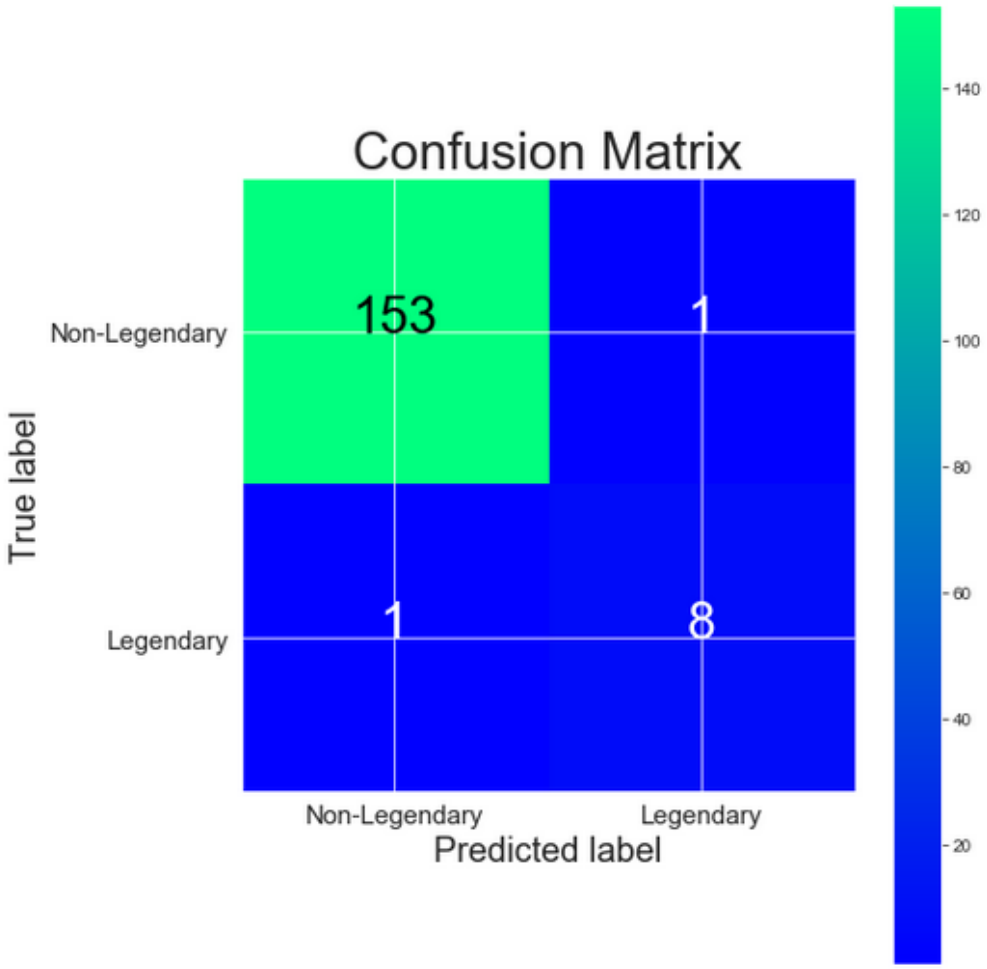


Figure 2.2.2. Training Data Confusion Matrix

The confusion matrix confirms that 163 data points have been analyses. Of these, 161 were predicted correctly and 2 we predicted incorrectly.

Error rate for i in range (1, 50) is calculated and plotted in a chart:

```
errorRate = []

for i in range(1,50):

    knn = KNeighborsClassifier(K = i)
    knn.fit(X_train,y_train)
    predict_i = knn.predict(X_test)
    errorRate.append(np.mean(predict_i != y_test))
```

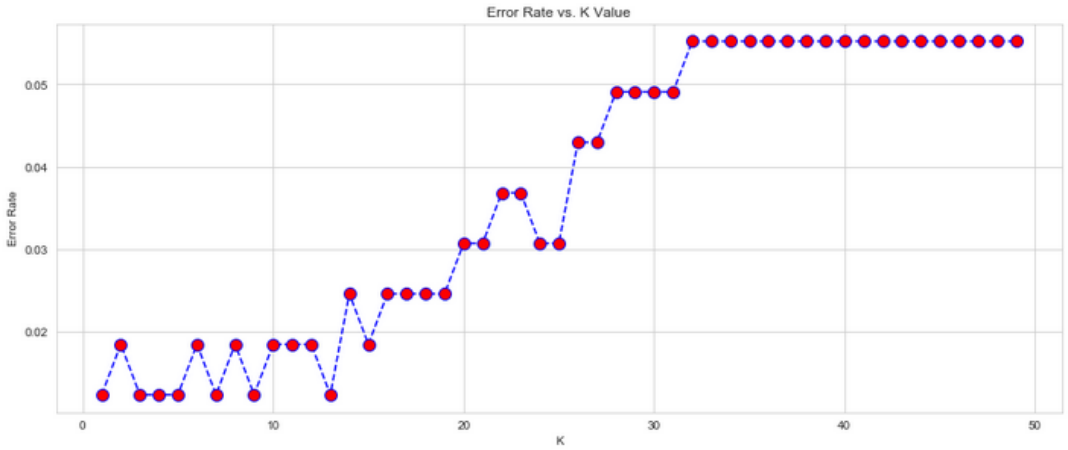


Figure 2.2.3. Error Rate vs K Value

Based on the chart, the error rate generally increases from a value of 13 to 32, where the error rate becomes constant. The predictor was run again with a value of K=13, prior to the increasing error rate trend.

```
knn= KNeighborsClassifier(K = 13)
knn.fit(X_train, y_train)
predictions = knn.predict(X_test)
```

```
accuracy = 0.988, precision = 0.988, recall = 0.988, f1 = 0.988
```

Changing the value of K to 13 did not affect the accuracy, which remained high at 98.8%. This supports the standard that using the square root of the number of data points is a suitable method for estimating K, as the square root of 163 (number of data points) is 12.7. The following table denotes the results from the other analyses, showing the change in the value of K.

2.2.5 Results

Analysis	K1	K2	Accuracy	Precision
1	5	13	98.8	98.8
2	5	9	98.8	98.8
3	5	14	96.3	96.3
4	5	3	96.3	95.9
5	5	9	95.7	95.9
6	5	7	98.2	98.3

Table 2.2.2. Analyses Accuracy Results

2.2.6 Discussion

The results confirm that some stats have a higher chance of indicating Legendary status than others. Analysis 1 and analysis 2 yielded the most accurate results. This indicates that weight and speed are given less consideration when determining if a Pokémon is legendary. Analysis 6, which considered all attributes, had a lower accuracy and precision score, indicating that some attributes may have a detrimental effect on classifying a Pokémon as Legendary.

3. Study 3: What are the strongest and weakest types of Pokémon (Bola & Bibi)

3.1 Objective

The aim is to find out the strongest and weakest types of Pokémon in the current dataset

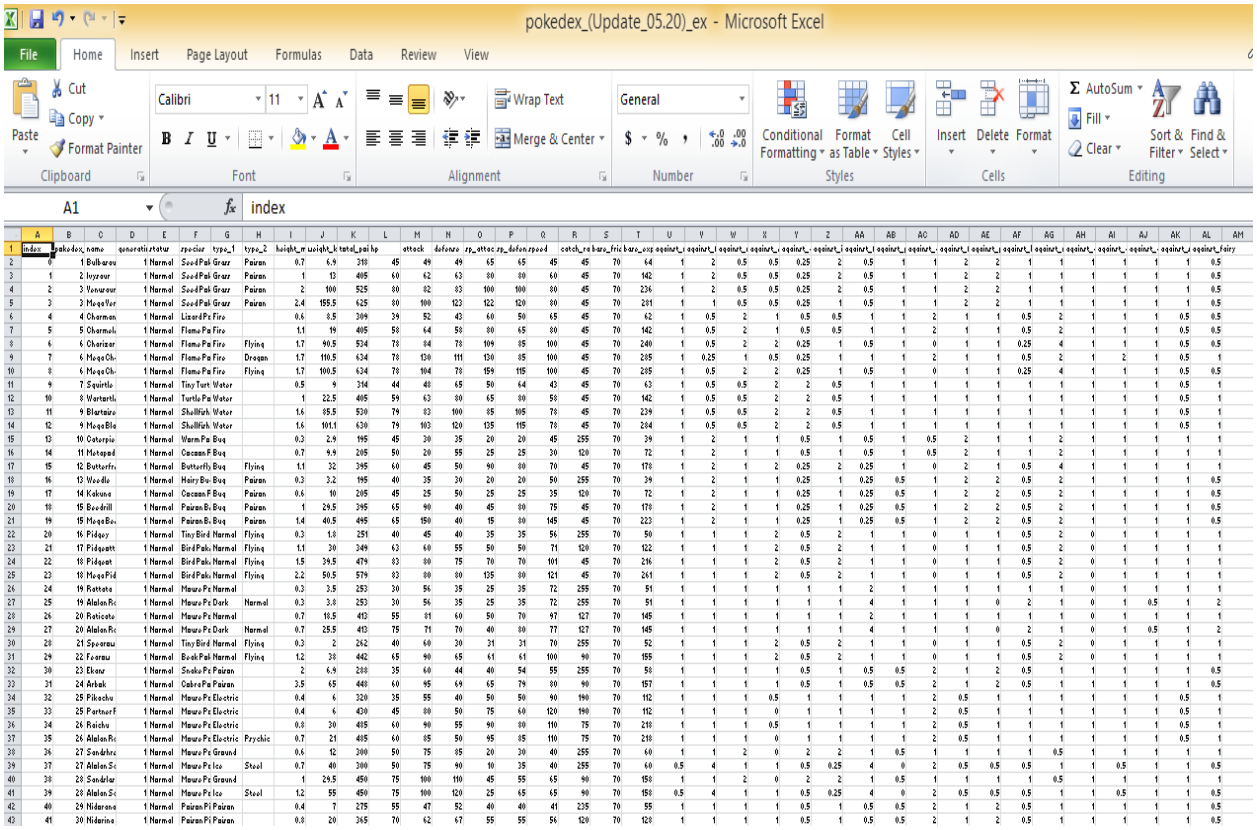
3.2 Problem Background

The Pokémon game revolves around a fictional species of collectable Pokémon each having unique design and skillsets. A player must build a team of Pokémon and battle them with other players Pokémon in-order to determine which player is the winner. As such, the simplest winning strategy is to ensure that a player has the strongest and most resilient team of Pokémon in-place before engaging in any battle. With this in mind, the objective of this analysis is to provide insight into which species of Pokémon can be classified as strong/weak with the goal of assisting players to target the right type of Pokémon when building their team.

3.3 Data Specification

Romero (2020) contains descriptive and quantitative information for 1028 Pokémon Pokémon. It describes the species/types and battle performance for each Pokémon by providing 37 attributes/features. Also, it has 641species. Below is a summary of these features:

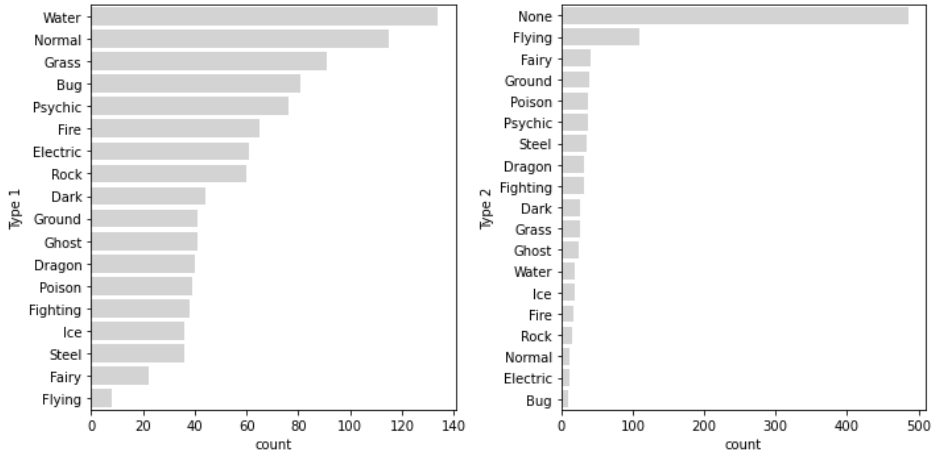
- pokedex_number - pokedex entry number of the Pokémon
- name - name of the Pokémon
- generation - refers to which grouping/game series the Pokémon was released in.
- status - a boolean that identifies whether the Pokémon is legendary
- species - species of the Pokémon
- type_1 - each Pokémon has a type, this determines weakness/resistance to attacks [referred to as the primary type]
- type_2 - some Pokémon are dual type and have 2 [referred to as the secondary type]
- height_m -
- weight_kg -
- total_points - sum of all stats that come after this, a general guide to how strong a Pokémon is
- hp - hit points, or health, defines how much damage a Pokémon can withstand before fainting
- attack - the base modifier for normal attacks
- defense - the base damage resistance against normal attacks
- sp_attack - special attack, the base modifier for special attacks
- sp_defense - the base damage resistance against special attacks
- speed - determines which Pokémon attacks first each round
- catch_rate - stronger Pokémon are generally harder to catch/collect
- base_friendship -
- base_experience - higher base stats indicate a better Pokémon
- against_X - determines a Pokémon's resistance against a specific type of Pokémon i.e, X can take the values fire, water, electric etc.



3.4 Research Questions

3.4.1 What are the most common types of Pokémon?

Common types of pokemon

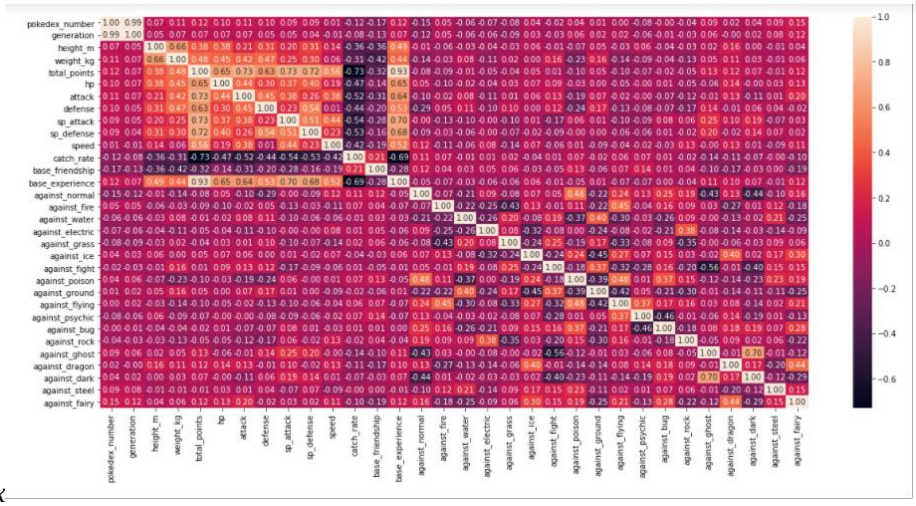


We can draw a few conclusions from these plots:

- All Pokémon species have a primary type.
- Normal-type, Water-type, and Grass-type Pokémon are a very common primary type. However, Normal-type Pokémon are outnumbered by Water-type Pokémon.
- Majority of Pokémon do not have a secondary type with Flying-type Pokémon being the most common secondary type.

Correlation Matrix

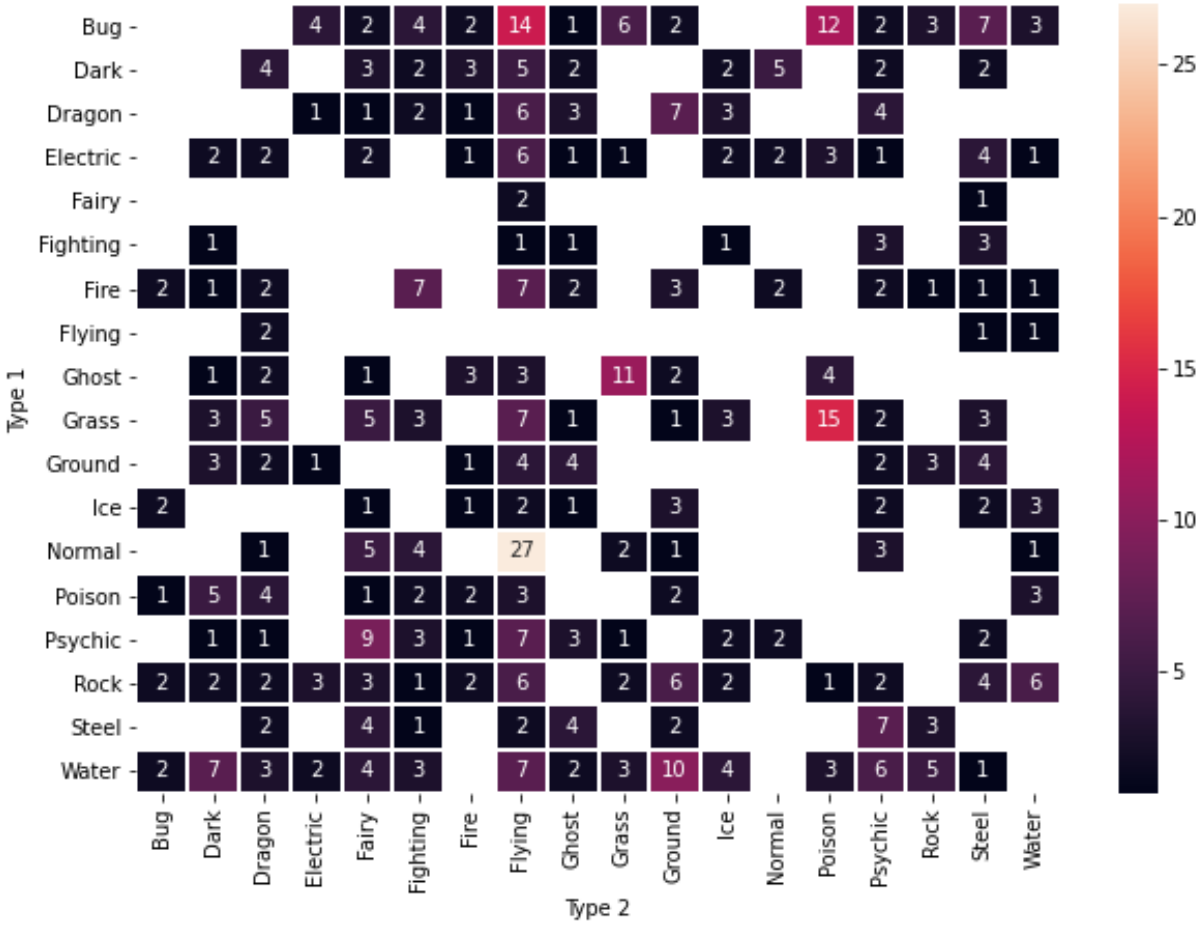
Complete correlation



Matrix

Combination of Primary and Secondary types.

We can investigate the distribution of the various combinations of primary and secondary types of Pokémon.



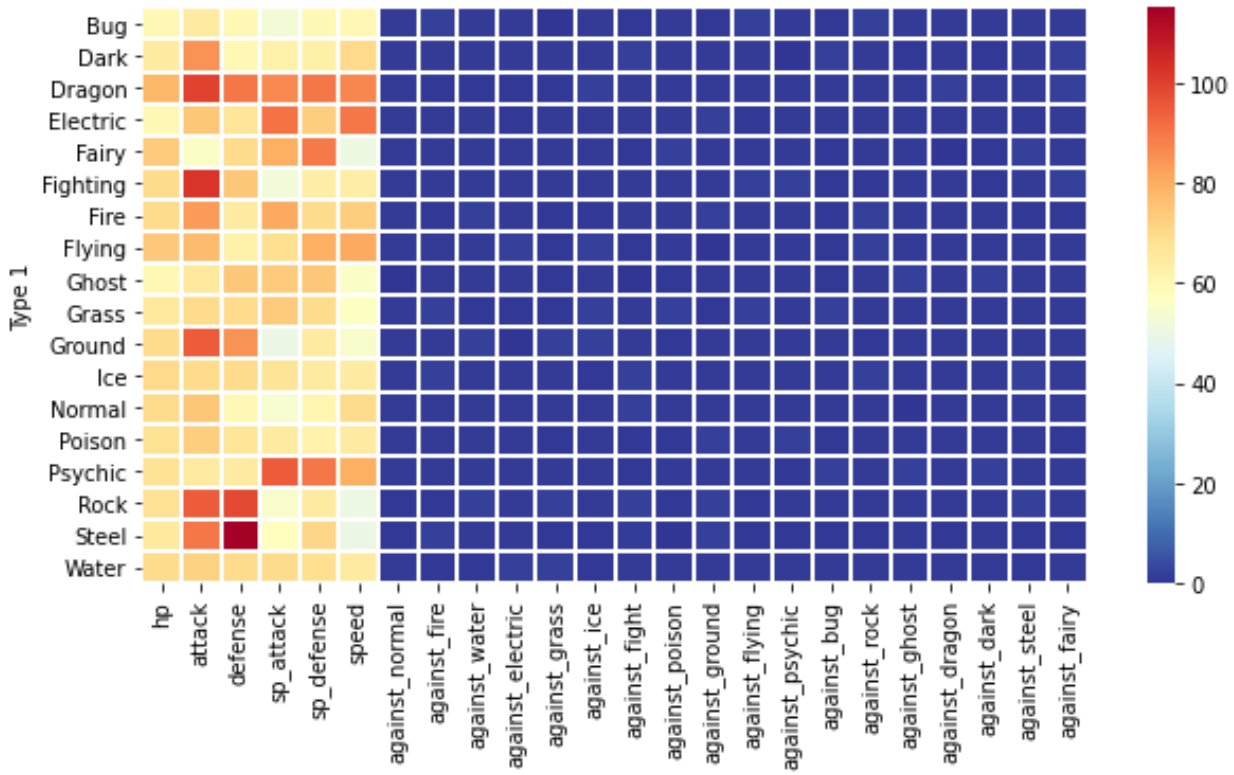
This plot reveals that five most common combinations of primary and secondary type are in order:

- Normal/Flying-type
- Grass/Poison-type
- Bug/Flying-type
- Bug/Poison-type
- Ghost/Grass-type
- Water/Ground-type

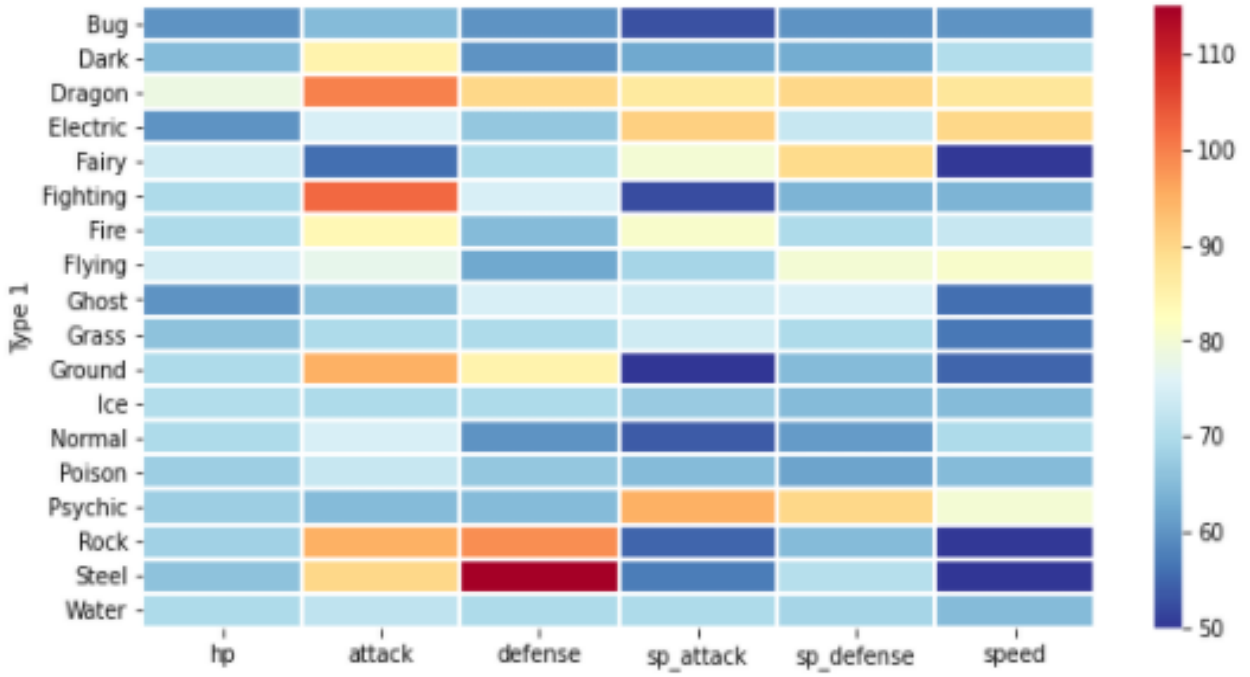
It is paradoxical; there exist Pokémon species that are Fire/Water-type and Ground/Flying-type

3.4.2 Do some types of Pokémon excel in some statistics over others?

We investigate if any specific statistic can influence how strong/weak a particular Pokémon is. Since all Pokémon have a primary type, we restrict our analysis to primary types only. Below are the statistics we assume determine strength/weakness.



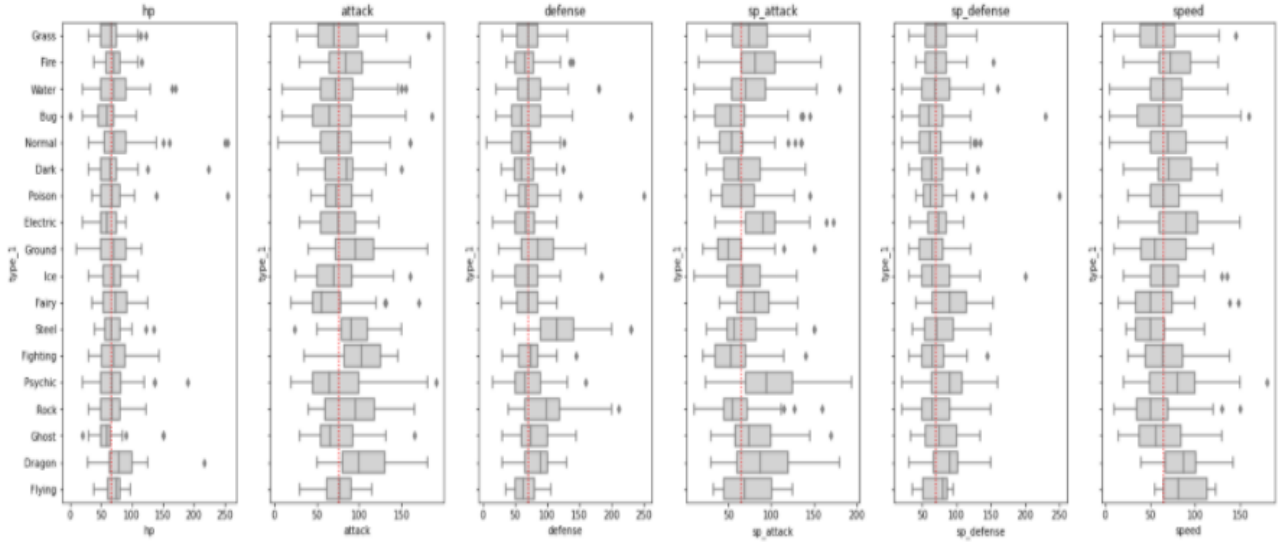
We observe that hp, attack, defense, sp_attack, sp_defense, and speed show the largest variability between the different types of Pokémon. Hence certain types of Pokémon do excel in these statistics over other types. As such, we restrict our analysis to these 6 features.



We can draw a few conclusions from these data:

- Pokémon with the highest defense in order are: Steel-type, Rock-type, Dragon-type, and Ground-type Pokémon.
- Pokémon with the highest attack in order are: Fighting-type, Dragon-type, Ground-type, and Rock-type Pokémon
- Flying-type Pokémon are relatively faster than Rock/Steel-type Pokémon

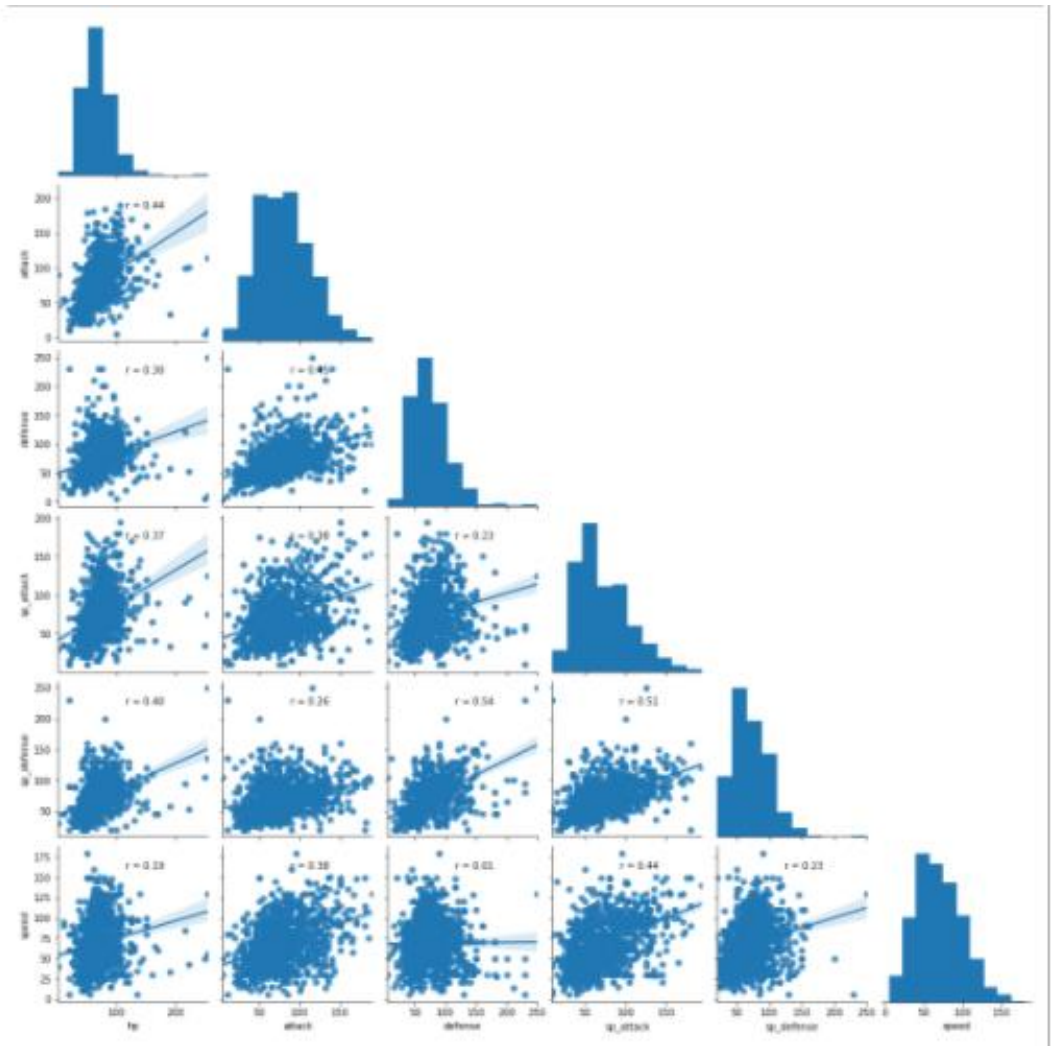
These results are intuitive. Moreover, by plotting the distribution of these statistics per Pokémon type, we can compare the results relative to the median of all Pokémon as shown in the graph below.



3.4.3 Are any of the statistics correlated?

Our findings have hinted that certain types of Pokémon which excel at a particular statistic don't fare so well in other statistics. To determine if the statistics are correlated, let's define a function that will

compute Pearson's correlation coefficient for each pair of statistics as a measure of the goodness of fit of a linear regression model.



This plot shows that the five most correlated pairs of statistics are in order:

- sp_defense / defense
- sp_defense / attack
- defense / attack
- attack / hp and sp_attack / speed
- sp_defense / hp

However, none of the statistics are strongly correlated, which is important as that would favour certain Pokémon or types of Pokémon over others. Furthermore, note that all the statistics are positively correlated except for defense and speed - this can be explained by considering Rock-type and Steel-type Pokémon.

Note that the above plot also reveals a few outliers with certain statistics that are significantly high. As such, we recommend handling these when doing further analysis.

3.5 Experiment: What are the strongest and weakest types of Pokémon?

3.5.1 K-Means Clustering

K – Means is a partitional clustering algorithm and a vector quantization method, this is considered the most common clustering algorithm because of its good balance between simplicity and efficacy. The method of partitioning data samples into k clusters follows a data partitioning strategy, such that the number of distances (or quantization error) between the data points and their corresponding cluster is determined (karthikeyan, 2020, p. 12).

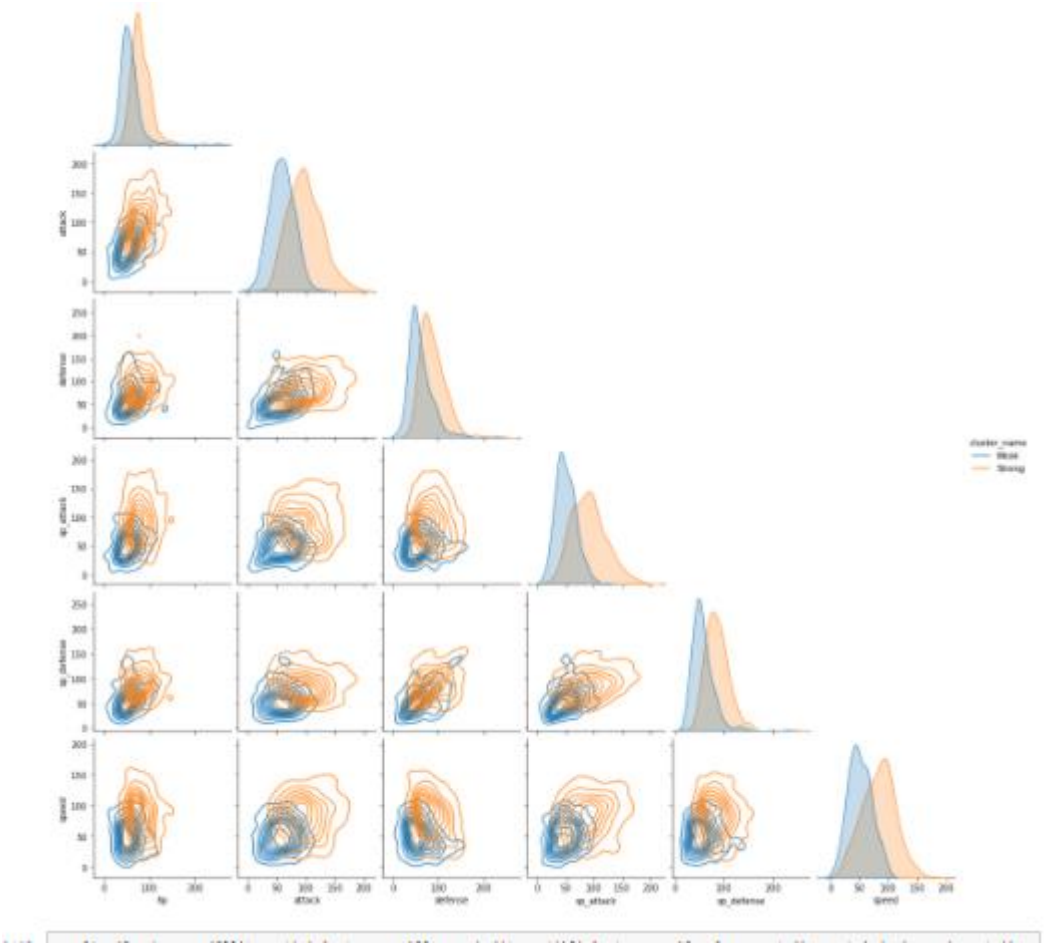
Our exploratory data analysis indicated that certain types of Pokémon excel in some statistic over others which could potentially indicate that some Pokémon are stronger than other Pokémon. As such we intend to use cluster analysis to partition the Pokémon into groups and analyse which Pokémon types can be clustered as strong and which Pokémon types can be clustered as weak.

In [40]:
```
kmeans = KMeans( init="random", n_clusters=2, n_init=10, max_iter=300, random_state=42)
kmeans.fit(X);
```

In [41]:
```
def cluster_names(kmeans):
    """Labels for cluster centers in Python sklearn

     e.g kmeans.cluster_centers_[0] = centroid of cluster 0
     See https://stackoverflow.com/questions/29460891/labels-for-cluster-centers-in-python-sklearn
    """
    c0 = kmeans.cluster_centers_[0]
    c1 = kmeans.cluster_centers_[1]

    cluster_map = {}

    if np.linalg.norm(c0)> np.linalg.norm(c1):
        cluster_map = {0:'Strong', 1:'Weak'}
    else:
        cluster_map = {1:'Strong', 0:'Weak'}

    return(cluster_map)

cluster_mapping = cluster_names(kmeans)
```

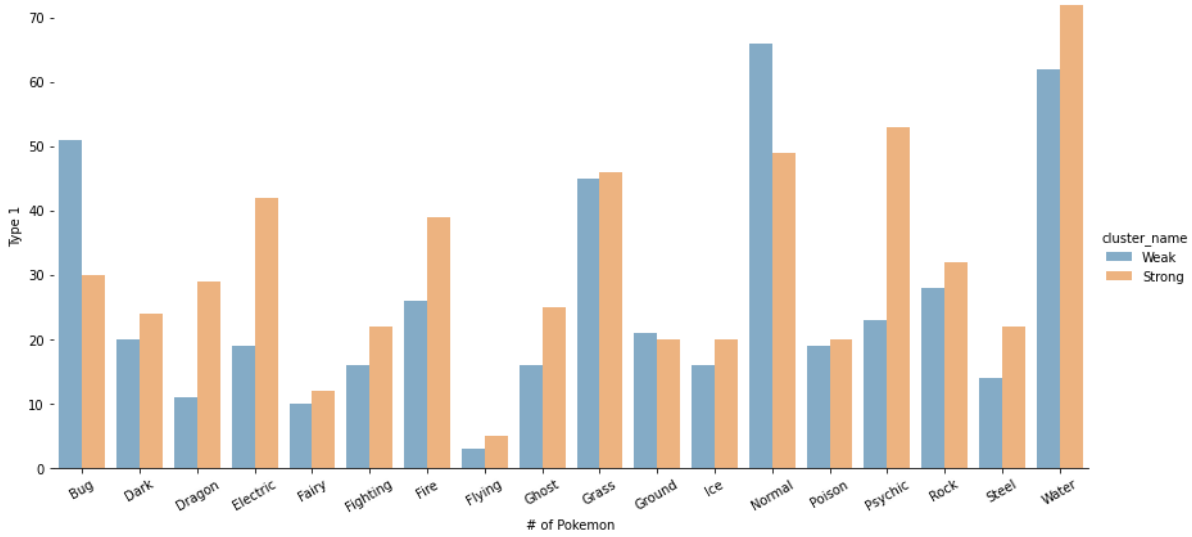
In [42]:
```
kmeans_df = df.copy()
kmeans_df['cluster'] = kmeans.labels_
kmeans_df['cluster_name'] = kmeans_df['cluster'].map(cluster_mapping)
```

In [43]:
```
sns.pairplot(
    data=kmeans_df[['type_1']+selected_features+['cluster_name']],
    kind='kde',
    hue = 'cluster_name',
    corner=True
);
```



In [44]:
```
results_df = kmeans_df[['type_1','cluster_name']].groupby('type_1')['cluster_name'].value_counts().reset_index(name='counts')
g = sns.catplot(
    data=results_df,
    kind="bar",
    x="type_1",
    y="counts",
    hue="cluster_name",
    ci="sd",
    alpha=.6,
    height=6,
    aspect =2
)
g.despine(left=True)
g.set_axis_labels("# of Pokemon", "Type 1")
g.set_xticklabels(rotation=30)
#g.legend.set_title("")
```

Out[44]:
```
<seaborn.axisgrid.FacetGrid at 0x85c6470be0>
```



Looking at the above we can see that:

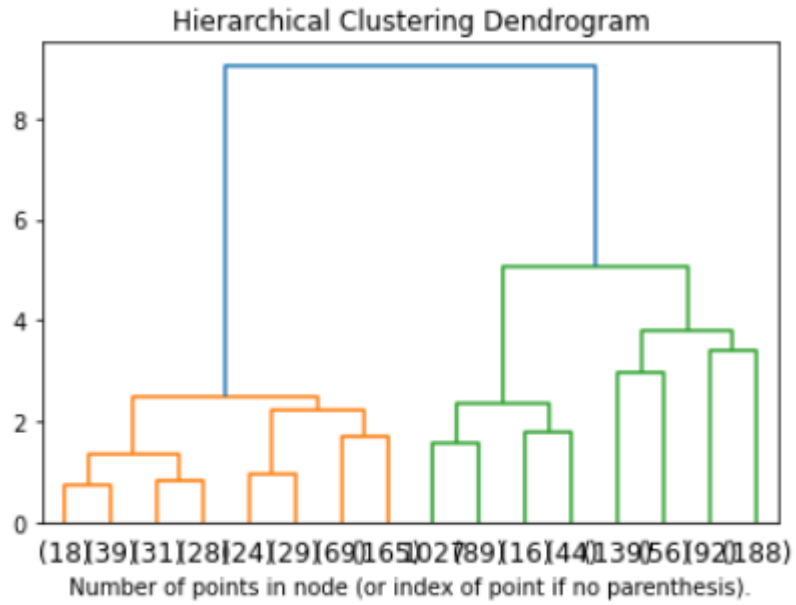
- ❖ Bug and Normal-type Pokémon are predominantly weak.
- ❖ Dragon, Electric, Fire, Psychic, Water, Fighting and Ghost-type Pokémon are predominantly strong.
- ❖ The remaining Pokémon-types are in between since there is no significant difference between the strong and weak types.

3.5.2 Hierarchal Clustering: Agglomerative Clustering

Hierarchical clustering is a method of cluster analysis which attempts to design a cluster hierarchy. To construct a hierarchy, two methods exist. The agglomerative approach is a bottom-up technique that merges clusters together gradually until a termination condition is met or a single cluster is left (karthikeyan, 2020, p. 14).

In [45]:
```
import numpy as np

from matplotlib import pyplot as plt
from scipy.cluster.hierarchy import dendrogram
from sklearn.cluster import AgglomerativeClustering

def plot_dendrogram(model, **kwargs):
    # Create linkage matrix and then plot the dendrogram

    # create the counts of samples under each node
    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1  # leaf node
            else:
                current_count += counts[child_idx - n_samples]
        counts[i] = current_count

    linkage_matrix = np.column_stack([model.children_, model.distances_,
                                      counts]).astype(float)

    # Plot the corresponding dendrogram
    dendrogram(linkage_matrix, **kwargs)
```

In [46]:
```
# setting distance_threshold=0 ensures we compute the full tree.
model = AgglomerativeClustering(distance_threshold=0, n_clusters=None)

model = model.fit(X)
plt.title('Hierarchical Clustering Dendrogram')
# plot the top three levels of the dendrogram
plot_dendrogram(model, truncate_mode='level', p=3)
plt.xlabel("Number of points in node (or index of point if no parenthesis).")
plt.show()
```



3.6 **Conclusion**

From the visual report analysis, the strongest and weakest Pokémon are determined by the six variables which are hp, attack, defense ,sp_attack, sp_defense and speed. We can see that the water type Pokémon are the strongest Pokémon and the flying type are the weakest Pokémon.

Overall Conclusion and Research Evaluation

Analyses on Pokémon data were performed and interesting results were found. Explorative clustering generated a new model to define groups of Pokémons. Classification system to identify legendary status of Pokémons was developed and tested. Lastly, strongest and weakest types of Pokémon were determined using correlation and clustering. These findings provided new information for Pokémon game players to search for suitable Pokémons and improve their in-game strategies. The research processes also enriched our understanding and skills on various data analytic methods and algorithms.

4. Peer Assessments

Daniel: 2.17

James: *2.17*

Leo: *2.17*

Vincent: 2.17

Bibi: 2.17

Bola: 2.17

5. References

Liberti, L., & Lavor, C., 2017, Euclidean Distance Geometry: An Introduction, Springer Undergraduate Texts in Mathematics and Technology, Springer.

Cohen, D., 2004, Precalculus: A Problems-Oriented Approach (6th ed.), Cengage Learning.

Lantz, B., 2013, Machine Learning with R, Packt Publishing, Limited, Olton.

Awad, M., & Khanna, R., 2015, Efficient Learning Machines : Theories, Concepts, and Applications for Engineers and System Designers, Apress L. P., Berkeley, CA.

Boehmke, B.C. and Greenwell, B. (2020) Hands-on machine learning with R. 1st edn.

Breiman, L. (2017) Classification and regression trees.

Gan, G., 2011, Data Clustering in C++ : An Object-Oriented Approach, CRC Press LLC, London.

Cichosz, P., 2015, Data Mining Algorithms : Explained Using R, John Wiley & Sons, Incorporated, Somerset.

Hackeling, G., 2014, Mastering Machine Learning with scikit-learn, Packt Publishing, Limited, Olton Birmingham.

Hwang, Y. H., 2018, C# Machine Learning Projects : Nine Real-World Projects to Build Robust and High-performing Machine Learning Models with C#, Packt Publishing, Limited, Birmingham.

Irizarry, R.A. (2019) Introduction to Data Science. Milton: CRC Press LLC.

Toomey, D., 2014, R for Data Science, Packt Publishing, Limited, Olton Birmingham.

Pierson, L., 2017, Data Science for Dummies, John Wiley & Sons, Incorporated, Somerset.

Kotu, V., & Deshpande, B., 2014, Predictive Analytics and Data Mining : Concepts and Practice with RapidMiner, Elsevier Science & Technology, San Francisco.

Dangeti, P., 2017. Statistics for machine learning. Packt Publishing Ltd.

Ward, J. H., Jr. (1963), "Hierarchical Grouping to Optimize an Objective Function", Journal of the American Statistical Association, 58, 236–244.

Kaufman, L., & Rousseeuw, P. J. (1990). Finding groups in data: Anintroduction to cluster analysis. New York: Wiley.

Park, H. S. and Jun, C. H., 2009. A simple and fast algorithm for K-medoids clustering. Expert systems with applications, 36(2), pp.3336-3341.

Ruspini, E. H., 1969. A new approach to clustering. Information and control, 15(1), pp.22-32.

Lloyd, S. (1957). Least squares quantization in pcm. Bell Telephone Laboratories Paper, Marray Hill.

Subramamanian, D., 2020. A Simple Introduction To K-Nearest Neighbors Algorithm. [online] Medium. Available at: <https://towardsdatascience.com/a-simple-introduction-to-k-nearest-neighbors-algorithm-b3519ed98e> [Accessed 6 November 2020].

Zhang, S., Li, X., Zong, M., Zhu, X. and Wang, R., 2018. Efficient kNN Classification With Different Numbers of Nearest Neighbors. IEEE Transactions on Neural Networks and Learning Systems, 29(5), pp.1774-1785.

karthikeyan, B. G. D. M. G. a. T. T., 2020. A Comparative Study on K-Means Clustering and Agglomerative Hierarchical Clustering. *International Journal*, 8(5), pp. 5-55.

Romero, M. T., 2020. https://www.kaggle.com/mariotormo/complete-pokemon-dataset-updated-090420?select=pokedex_%28Update_05.20%29.csv. [Online]