Group Members:
- Chase Moreno 2078503
- Daniel Laz

MATH 4323 Project: Final Report

1. Introduction (Chase Moreno).

We will be analyzing a dataset called "Pokemon with stats" which is available on Kaggle.com. In this dataset, it has 800 different types of Pokemon and contains 13 variables for each Pokemon. The 13 variables are their ID number, Name, Type1, Type2, Total sum of all stats, Hit Points (HP), Attack Strength, Defense Strength, Special Attack Strength, Special Defense Strength, Speed, Generation, and Legendary. Our main overall question of this analysis is "Can we determine any patterns or relationships between various Pokemon species based on their stats and types?" To do this we will use KMeans clustering and Hierarchical clustering algorithms to find the similarities and differences between Pokemon based on their stats by grouping. Before we start doing any algorithms, we like to explore the dataset to check if there are any missing values and get the basic understanding for each variable. There are 18 types of Pokemon such as bugs, fire, poison, steel, water, and so on. For type1 features, most pokemon are water type which is 112 of them and the least type of pokemon is flying with only 4 pokemon. On the other hand, there are many flying types of pokemon for type2 features which are 97 of them and there are few bug types of pokemon which are only 3. However, we have 386 missing values for type2 feature because these 386 pokemon have an empty value for this specific column. The way we fix this issue is that we decided to replace these empty values with the value as "None". It will help our dataset to stay cleaner than having missing values that can lead to inaccurate results. We can always ignore the type2 variable if we are uncomfortable with using it in our model when finding the patterns between various Pokemon based on their types. There are only 6 generations and there are three generations that have over 160 pokemons. For instance, 166 pokemon in generation 1, 160 pokemon in generation 3, and 165 in generation 5. A generation refers to a pokemon that was first introduced to a particular series of Pokemon games. There 65 legendary pokemon in this dataset. All type1 pokemon do have legendary pokemon except bug, fighting, and poison pokemon type. We discovered that the strongest pokemon is MewtwoMega Mewtwo X which has a total of 780 stats from adding all base stats. Of course, it is a legendary pokemon and it is a psychic - fighting type of pokemon. We also discovered the weakest pokemon and it is Sunkern which has a total of 180 stats. This pokemon is not legendary and is a grass type of pokemon.

2. Methodologies (Daniel Laz).

The two distinct models we will be using are KMeans and Hierarchical clustering. The first model will be KMeans clustering is known as unsupervised learning that is partitioned data set into K (non-overlapping) clusters. Also for every data point is appointed to the clusters whose mean is nearest to the point. The means of each cluster are then being recomputed and data points are reassigned to the nearest mean. And this cycle is rehashed until the means never again move essentially or a most extreme number of emphasis is reached. One of the goals for KMeans is that it limits the WCSS (Within-Cluster Sum of Squares) distances between each point and its appointed group mean. For example, we will use the elbow method to determine the optimal K clusters and it will help to limit the WCSS. The expected potential advantages of K-means is that

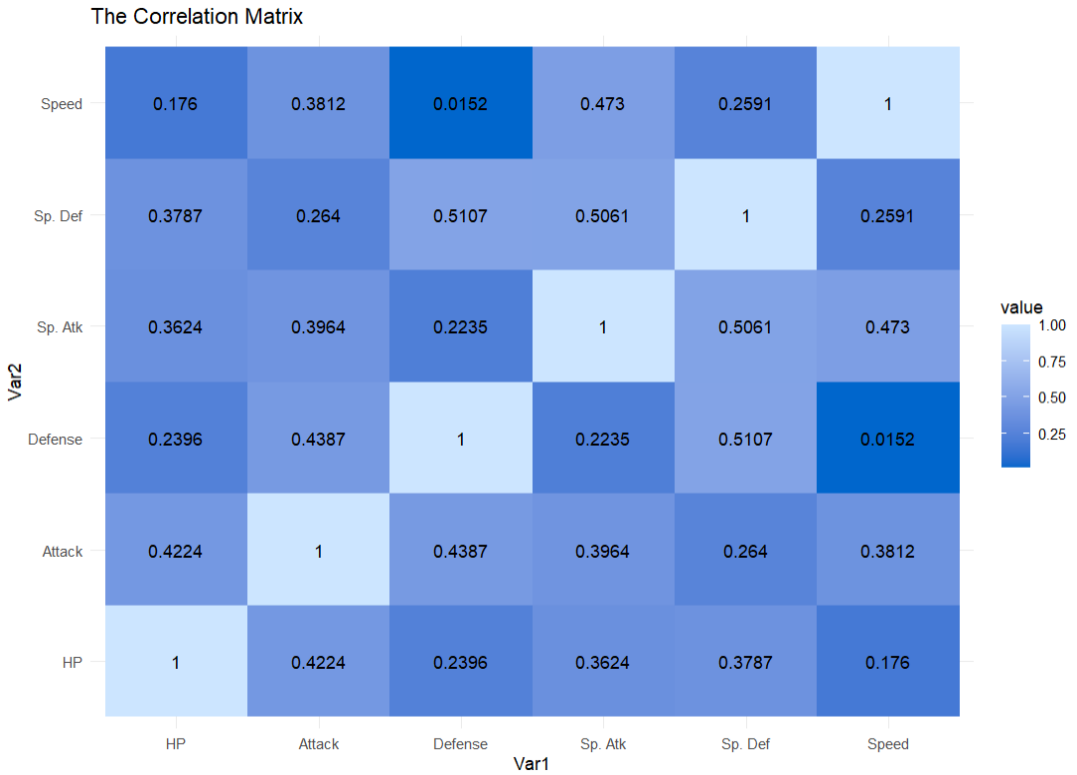
it can eventually scale the large data set, is easy to comprehend and shows the easy way to explain the clusters, and form the clusters to different sizes or shapes. The disadvantages of KMeans is the need to pre-specify the number of clusters K, which can be a difficult task to decide and can impact the effectiveness of clusters. The main optimization criteria of KMeans is to limit the sum of squared distances between each point and its relegated group centroid, otherwise called Within-Cluster Sum of Squares (WCSS). The formula for WCSS is as per following:

$$\text{Minimize} \quad \{\sum_{k=1}^{K} \quad \frac{1}{|C_k|} \sum_{i,j \in C_k, i>j} \quad \sum_{r=1}^{p} \quad (X_{ir} - X_{jr})^2\}. \\ C_1, \ldots, C_k$$

The second model will be hierarchical clustering, which is known as unsupervised learning for clustering the data points we want to cluster them into groups based on their similarity. When beginning the process of algorithms, every observation is assigned its own groups, and then the algorithm proceeds by iteratively consolidating the nearest sets of groups until a halting rule is met, like arriving at an ideal number of groups or a specific degree of comparability between groups. The other factor of hierarchical clusters is that distance metric or similarity measure utilized to compute the distance among groups, and also the linkage techniques used to choose how to consolidate clusters. We can find the optimal number of clusters by using the silhouette coefficient where the highest value is the best. For hierarchical clustering, the advantage is that it is simple to comprehend and execute. It also provides the tree-like visual representation of the clusters which is the dendrogram to help us comprehend to see the picture of the group in the data. Also, it does not require the number of clusters to be specified in advance. The disadvantage for hierarchical clustering is that it can be slow for clustering particularly when it comes to large datasets with a large number of observations. There are different linkage methods such as single, complete, average, and centroids. For single linkage is to find the minimum distance between the nearest points of two groups. Complete linkages is to find the clusters based on the maximum distance between the farthest points of two groups. The average linkage is to find the average distance between all points in two clusters. Lastly, centroid linkage is to find the distance between the two clusters by computing the distance between the centroid of the clusters.

3a. Data Analysis (Chase Moreno, Daniel Laz).

Refer to our main overall question in this analysis, “Can we determine any patterns or relationships between various Pokemon species based on their stats and types?” First, we need to exclude predictors that are categorical variables which are Name, Type1, Type2, and Legendary. Also, we decided to remove the ID number, The total sum of all stats, and Generation because it is not relevant to what we need to determine the patterns between various pokemon. So that way we will have an accurate result and good modeling performance. Our chosen features for data analysis are HP, Attack, Defense, SP Attack, SP Defense, and Speed. We created a heatmap of the correlation matrix based on these features.

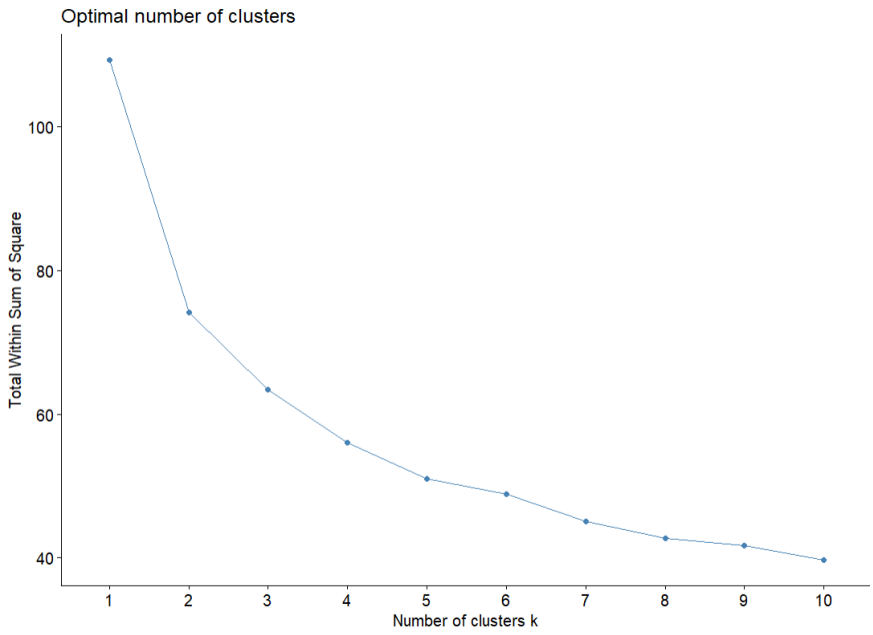


According to the heatmap shown above, we do not have any severe collinearity because there is no correlation with more than 0.8 between 2 variables. We have two highly correlated relationships which are Defense & Sp. Defense with a value of 0.5107 and Sp. Attack & Sp. Defense with a value of 0.5061. Also, we have one lower correlation relationship which is Defense and Speed with a number of 0.0152. Now we need to scale the data to keep all variables having the same influence in the clustering process. So, we will do the min-max scale where it will put the values to a range between 0 and 1. Using library(scales), we scale our data to [0 - 1]:

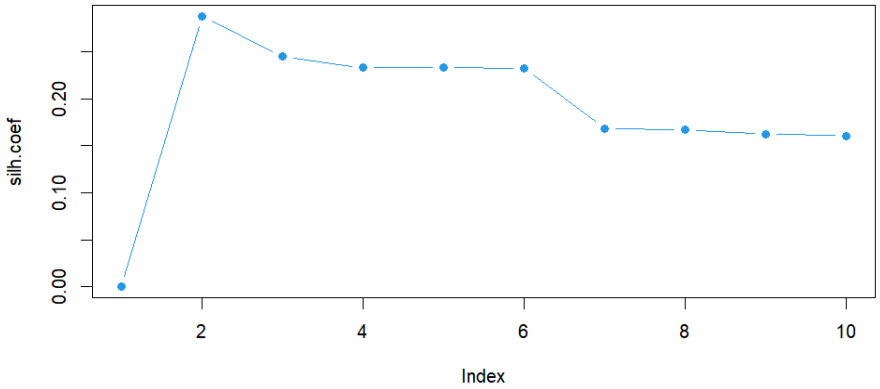
```
> # Scale the data using min-max scaling way
> library(scales)
> scaled_data <- as.data.frame(sapply(features_data,
+                  function(x) rescale(x, to = c(0, 1))))
>
> # Call head to return first 5 data values
> head(scaled_data)
         HP    Attack   Defense   Sp. Atk   Sp. Def     Speed
1 0.1732283 0.2378378 0.1955556 0.2989130 0.2142857 0.2285714
2 0.2322835 0.3081081 0.2577778 0.3804348 0.2857143 0.3142857
3 0.3110236 0.4162162 0.3466667 0.4891304 0.3809524 0.4285714
4 0.3110236 0.5135135 0.5244444 0.6086957 0.4761905 0.4285714
5 0.1496063 0.2540541 0.1688889 0.2717391 0.1428571 0.3428571
6 0.2244094 0.3189189 0.2355556 0.3804348 0.2142857 0.4285714
```

3b. Data Analysis (Chase Moreno, Daniel Laz).

We can start performing our model for Kmeans and Hierarchical clustering. Before we do anything else, we should find the optimal tuning parameter value for our model to have a good performance. Let's begin with KMeans of finding the optimal K value using the elbow method.



According to the figure above, it is difficult to select the optimal K value from this elbow method because we hardly see any elbow point. It can be k = 2, but we believe it is better to calculate the silhouette coefficient for the optimal clustering because we still want to be sure that we have the optimal K values before fitting the model. Here is the chart where we calculate and visualize average silhouette coefficients across KMeans solutions for multiple K:



It seems k = 2 has the largest silhouette coefficient. Here is the silhouette coefficient for k = 2 due to large amount of silhouette coefficient report we just going to include the average of number of clusters:

```
$clus.avg.widths
[1] 0.1838268 0.4050383

$avg.width
[1] 0.2877962
```

We can move onto Hierarchical for finding the optimal K using various linkages methods like single, complete, and average linkages.

The silhouette coefficient for single linkage:

```
> print(hc.sing$silinfo$avg.width)
[1] 0.5075546
```

The silhouette coefficient for complete linkage:

```
> print(hc.comp$silinfo$avg.width)
[1] 0.2410721
```

The silhouette coefficient for average linkage:

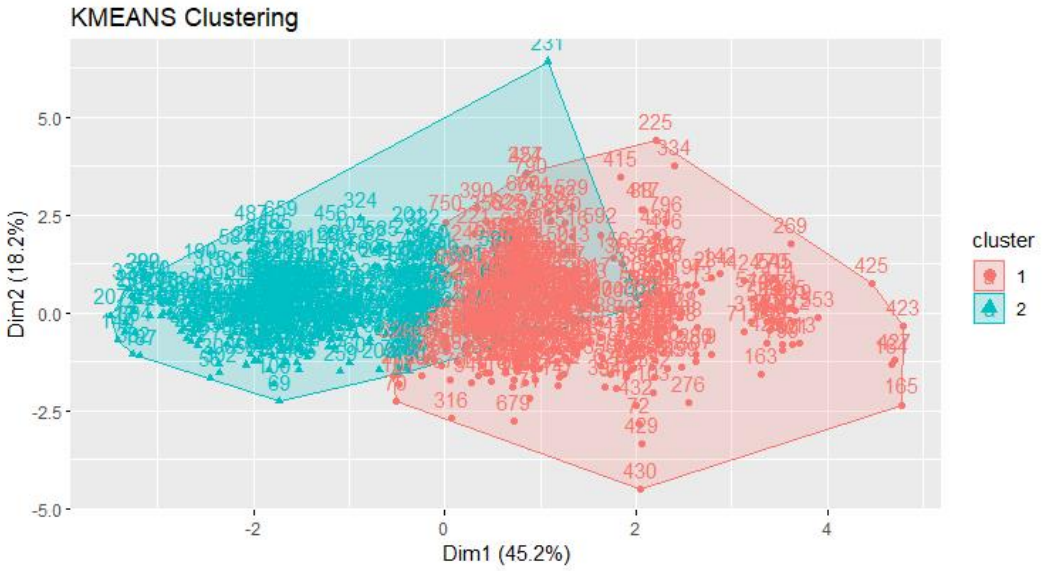
```
> print(hc.avg$silinfo$avg.width)
[1] 0.5075546
```

3c. Data Analysis (Chase Moreno, Daniel Laz).

Now we have calculated the silhouette coefficients for both models. It is time to compare and select the optimal cluster based on the highest silhouette coefficient for KMeans and Hierarchical clustering. For KMeans we had found that k = 2 is optimal due to the highest silhouette coefficient which is 0.2877962. The silhouette coefficient for cluster 1 is 0.1838268 and 0.4050383 for cluster 2. For Hierarchical clustering we have the average silhouette coefficient of 0.5075546 for single linkage, 0.2410721 for complete linkage, and 0.5075546 for average linkage. It turns out that single and average linkage has the highest silhouette coefficient and we decided to choose average linkage when it is time to fit our model.

3d. Data Analysis (Chase Moreno, Daniel Laz).

We begin to fit our model using the best silhouette coefficient that we found. Here is KMeans clustering scatter plot from using eclust() function:

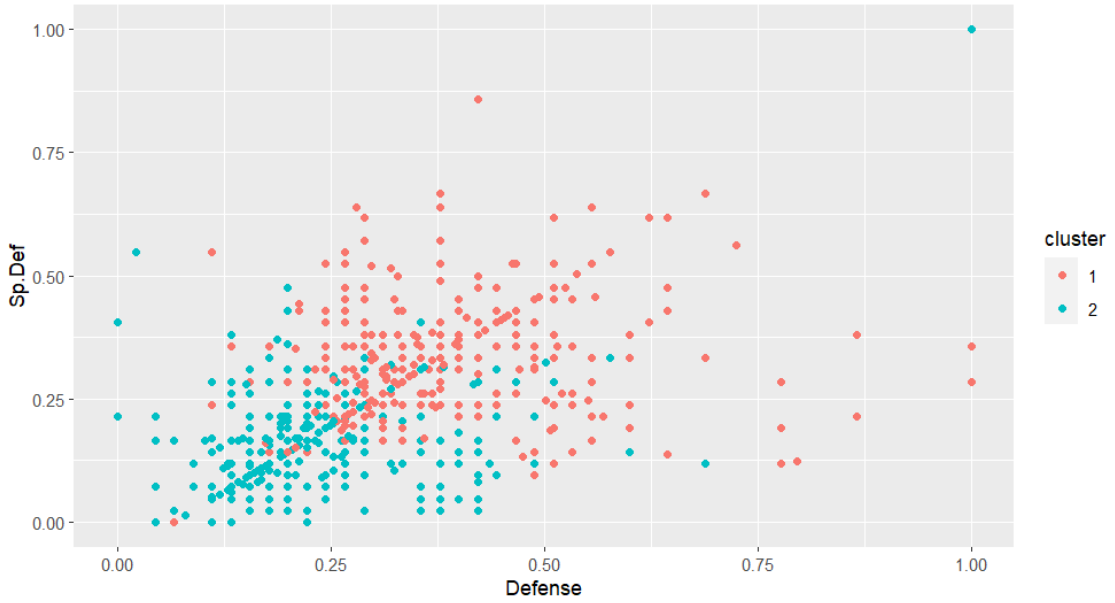


The model summary:

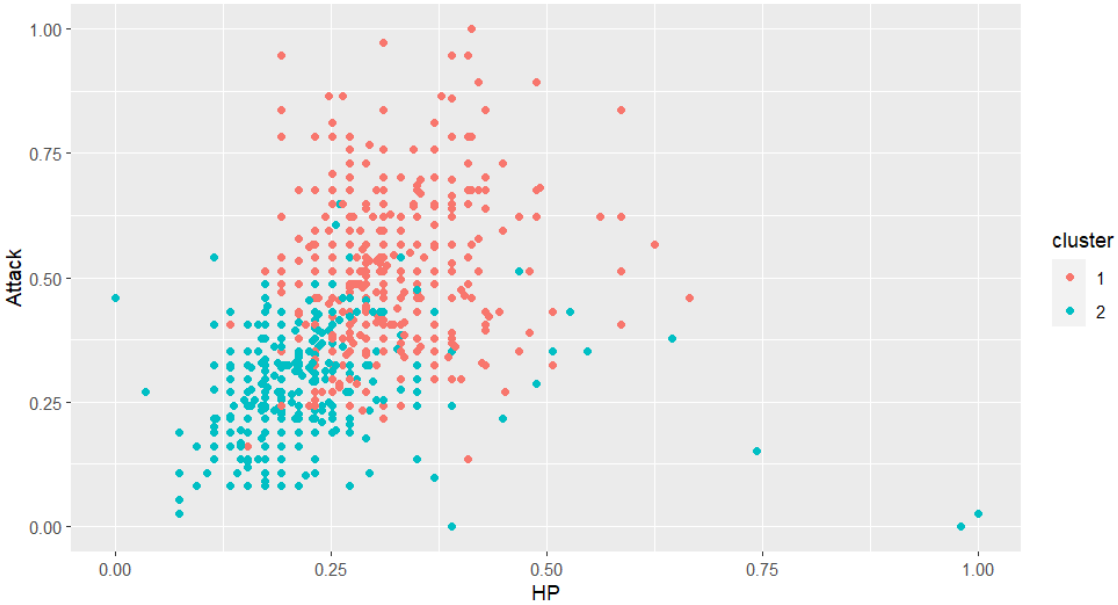
```
> summary(km.model)
             Length Class      Mode
cluster      800    -none-     numeric
centers       12    -none-     numeric
totss          1    -none-     numeric
withinss       2    -none-     numeric
tot.withinss   1    -none-     numeric
betweenss      1    -none-     numeric
size           2    -none-     numeric
iter           1    -none-     numeric
ifault         1    -none-     numeric
clust_plot     9    gg         list
```

```
silinfo         3    -none-     list
nbclust         1    -none-     numeric
data            6    data.frame list
```

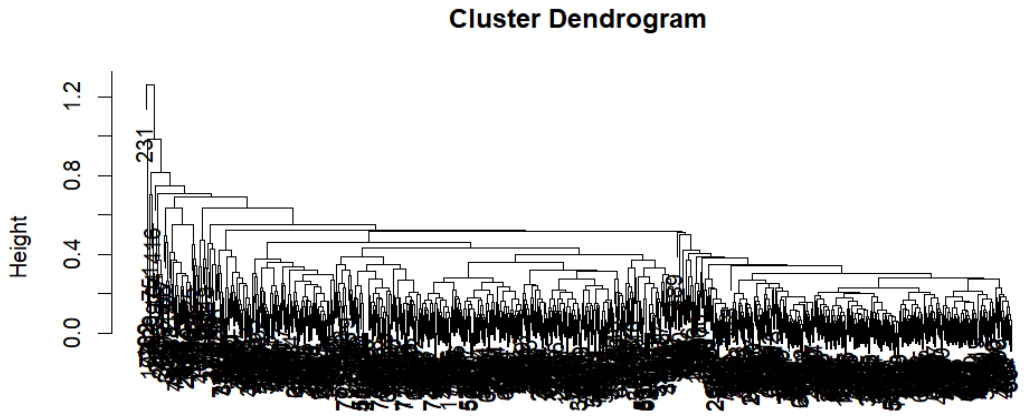
Using ggplot2, here is a scatterplot of Defense vs. Special Defense:



Scatterplot of HP vs. Attack:

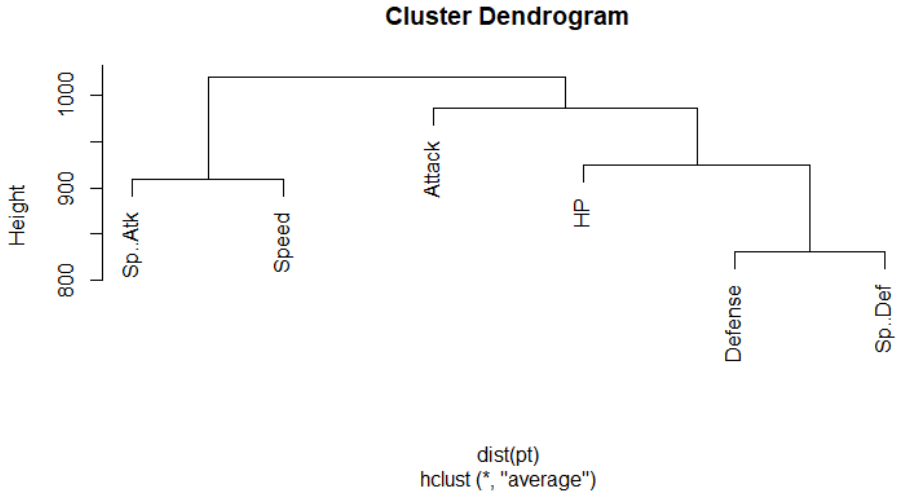


Here is the dendrogram for Hierarchical clustering using average linkage:




```
> summary(hc.avg)
            Length Class  Mode
merge       1598   -none- numeric
height       799   -none- numeric
order        800   -none- numeric
labels         0   -none- NULL
method         1   -none- character
call           3   -none- call
dist.method    1   -none- character
```

Since, our dendrogram above is hard to visualize due to large amounts of observations. So, here is the dendrogram of the features down below that can help us to visualize what observations are similar and differ to each other.



3e. Data Analysis (Chase Moreno, Daniel Laz).

Refer to our main overall question in this analysis, “Can we determine any patterns or relationships between various Pokemon species based on their stats and types?” Based on the question from this analysis, we can determine the relationships between the various Pokemon species based on their stats and types. From looking at the correlation matrix, we can see two

highly correlated relationships which are the Defense and Sp. Defense with about 51 percent and Sp. Attack and Sp. Defense with about 50 percent. This is a good sign because it showed patterns where these features are similar to each other. We began to use clustering models such as KMeans and Hierarchical in order to cluster the Pokemon species into groups to see the comparison between different models to see the results. From the KMeans model, we use the elbow method to figure out which optimal K value and guessing the optimal K value is 2. But that wasn't enough proof for us to know the actual optimal value. So we decided to find the silhouette coefficient for each cluster and it shows k = 2 is the highest because the silhouette score is 0.2877962. The Hierarchical model, we find the highest silhouette score from average linkage which is 0.5075546 and we are using the average linkage to fit our model. We fit our best model for KMeans and plot the clusters using the eclust() function. As you can see from part d, we have two clusters where cluster 1 is clustered on the right side and cluster 2 is on the left side. However, we need more information to determine any patterns between various Pokemon. We can see that some observations of cluster 1 appear to be bad because they are already in cluster 2 for instance, observations; 750, 390, 790, 221, 628, 240, 660 and much more. While we have other observations like 316, 679,70, and 415 are close to cluster 2. The cluster 2 appears to be bad as well because some of the observations are in cluster 1. Furthermore, we plotted a scatter of Defense vs. Special. Def which are highly correlated with each other. We see that there are Pokemon that have both high Defense and Special Defense strength compared to other pokemon. We also, plot the scatter plot of HP vs. Attack and it shows that many Pokemon with weak attack strength often have low HP. However, there are few that have higher HP than the majority of Pokemon with decent attack strength. Overall, cluster 1 still has the best values due to HP and Attack at most of the time. Refer back to our eclust() plot, we can now say the Pokemon from cluster 1 are stronger and more powerful than the Pokemon from cluster 2. From the hierarchical section, we plot a dendrogram and it is hard to see which observations are being grouped. We decided to add another dendrogram for us to see clearly on which features that may relate to and the comparison of which clusters the features being assigned to. In the plot, we can decide to make a horizontal cut, for the height at 1000 to have two clusters. So, we have Special Attack and Speed in one cluster. For cluster 2, we have Attack, Hit Points, Defense, and Special Defense. We can see the Pokemon with Special Attack strength and Speed are similar to each other as well as Defense and Special Defense. We can see that Pokemon with good Attack strength are not quite similar to Hit Points as they are to Defense and Special Defense.

4. Conclusion (Chase Moreno, Daniel Laz).

We determined patterns and relationships various between Pokemon by their stats such as HP, Attack, Defense, SP. Attack, SP. Defense, and Speed. Our KMeans clustering algorithm using k = 2 created two clusters where Pokemon from cluster 1 are stronger and Pokemon from cluster 2 are weaker. We do have some difficulties from our KMeans model because we have outliers in cluster 2 that created cluster 2 to overlap cluster 1. We could use different k values, but k = 2 is our optimal clustering due to the highest silhouette coefficient. We believe there are possible procedures to improve this KMeans model by removing the outliers before fitting the model. We know that KMeans is very sensitive to outliers and we should have checked the outliers in our initial exploratory analysis. Based on the hierarchical model, we tend to have some difficulties on the dendrogram tree whereby we are not able to see which Pokemon are similar to each other due to cluttered branches and leaves. We believe using PCA could help to

fix this issue to reduce the data. Also, we could eliminate a certain number of observations or features in our subset to reduce the high number of data for plotting into the dendrogram.

References:

Kaggle.
https://www.kaggle.com/datasets/abcsds/pokemon

Lecture 10 ppt slides.
https://elearning.uh.edu/bbcswebdav/pid-9981733-dt-content-rid-88157896_1/courses/H_20231_MATH_4323_15666/MATH_4323_lect10.pdf

Lecture 11 ppt slides.
https://elearning.uh.edu/bbcswebdav/pid-9981734-dt-content-rid-88157898_1/courses/H_20231_MATH_4323_15666/MATH_4323_lect11.pdf

Lecture 12 ppt slides.
https://elearning.uh.edu/bbcswebdav/pid-9981735-dt-content-rid-88157900_1/courses/H_20231_MATH_4323_15666/MATH_4323_lect12.pdf

Lab 8 Material.
https://elearning.uh.edu/bbcswebdav/pid-9981715-dt-content-rid-88157866_1/courses/H_20231_MATH_4323_15666/MATH_4323_lab8.pdf

Appendix (Daniel Laz, Chase Moreno)

3a (The correlation matrix)
Rcode:

```
> # Create the correlation matrix
> cor_mat <- cor(features_data)
>
> # Reshape the correlation matrix
> cor_mat_melt <- melt(cor_mat)
>
> # Create a heatmap of the correlation matrix
> ggplot(data = cor_mat_melt, aes(x = Var1, y = Var2, fill = value)) +
+  geom_tile() +
+  scale_fill_gradient(low = "#0066CC", high = "#CCE5FF") +
+  theme_minimal() +
+  labs(title = "The Correlation Matrix") +
+  geom_text(aes(label = round(value, 4)), color = "black")
```

3b (KMeans using elbow method)
Rcode:

```
> # Find the optimal k value using the elbow method
> library(factoextra)
> kclust <- sapply(1:10, function(k) {
+  km.model <- eclust(scaled_data, FUNcluster = "kmeans", k = k,
+                     nstart = 50)})
> elbow <- fviz_nbclust(scaled_data, FUNcluster = kmeans, method = "wss")
> elbow
```

3b (KMeans using the silhouette Method)
Rcode:

```
> # Find the silhouette Coef. for all k values
> k.max <- 10
> km.silh <- numeric(k.max)
> for (k in 2:10){
+  km.silh[k] <- eclust(scaled_data, FUNcluster = "kmeans", k = k, graph = 0,
+                       nstart = 50)$silinfo$avg.width
+ }
> plot(km.silh, type = "b", pch = 19, col = 4)
```

3b (KMeans Silhouette Coef. calculations for k = 2)
Rcode:

```
> # Create the for optimal k=2 values
> km.model <- eclust(scaled_data, FUNcluster = "kmeans", k = 2, nstart = 50)
> # To print out the silhouette coef.
> km.model$silinfo
$widths
    cluster neighbor  sil_width
718       1        2 0.41180507
697       1        2 0.40756594
709       1        2 0.40721390
166       1        2 0.40591836
```

272	1	2	0.40591836
428	1	2	0.40591836
549	1	2	0.40591836
551	1	2	0.40591836
554	1	2	0.40591836
8	1	2	0.40110619
793	1	2	0.39736980
794	1	2	0.39736980
367	1	2	0.39589920
398	1	2	0.39424719
553	1	2	0.39358869
162	1	2	0.39353413
708	1	2	0.39313236
539	1	2	0.39226809
409	1	2	0.39184875
711	1	2	0.39178156
542	1	2	0.38951663
159	1	2	0.38866788
707	1	2	0.38553664
541	1	2	0.38452754
410	1	2	0.38450420
710	1	2	0.38137901
65	1	2	0.37964087
4	1	2	0.37856797
158	1	2	0.37852486
704	1	2	0.37656036
703	1	2	0.37613026
705	1	2	0.37613026
13	1	2	0.37531177
420	1	2	0.37362283
421	1	2	0.37165004
546	1	2	0.37099372
414	1	2	0.36949090
426	1	2	0.36851852
494	1	2	0.36781280
424	1	2	0.36774400
543	1	2	0.36562056
495	1	2	0.36475100
800	1	2	0.36294358
271	1	2	0.36206830
249	1	2	0.36122308
163	1	2	0.35963202
540	1	2	0.35872618
284	1	2	0.35868819
263	1	2	0.35826226
264	1	2	0.35791132
280	1	2	0.35735986

419	1	2	0.35710769
712	1	2	0.35684636
499	1	2	0.35664554
552	1	2	0.35605411
422	1	2	0.35485991
9	1	2	0.35383897
550	1	2	0.35382540
544	1	2	0.35358717
795	1	2	0.35190023
250	1	2	0.35177861
518	1	2	0.35107585
413	1	2	0.34948561
706	1	2	0.34932062
701	1	2	0.34911218
268	1	2	0.34693687
713	1	2	0.34675696
418	1	2	0.34579386
646	1	2	0.34471129
545	1	2	0.34374836
797	1	2	0.33958719
438	1	2	0.33910194
350	1	2	0.33872527
157	1	2	0.33821279
629	1	2	0.33704326
717	1	2	0.33429401
519	1	2	0.33295709
7	1	2	0.33230794
172	1	2	0.33230794
714	1	2	0.33135080
715	1	2	0.33135080
498	1	2	0.33020526
276	1	2	0.32971193
716	1	2	0.32729857
269	1	2	0.32650685
279	1	2	0.32634813
24	1	2	0.32501860
702	1	2	0.32413126
164	1	2	0.32301051
270	1	2	0.32257798
142	1	2	0.32007459
125	1	2	0.31961266
425	1	2	0.31830192
777	1	2	0.31794636
427	1	2	0.31702730
700	1	2	0.31604792
314	1	2	0.31443536
423	1	2	0.31321104

798	1	2	0.30888231
451	1	2	0.30867422
394	1	2	0.30770258
138	1	2	0.30758647
528	1	2	0.30483893
799	1	2	0.30468635
727	1	2	0.30283956
362	1	2	0.30233242
265	1	2	0.30201069
175	1	2	0.30174477
165	1	2	0.30110679
275	1	2	0.29929876
155	1	2	0.29778888
3	1	2	0.29661013
307	1	2	0.29571115
563	1	2	0.29352521
340	1	2	0.29334799
388	1	2	0.28918136
337	1	2	0.28895264
724	1	2	0.28745434
230	1	2	0.28604551
526	1	2	0.28556178
633	1	2	0.28201221
682	1	2	0.27993140
283	1	2	0.27503314
12	1	2	0.27503099
131	1	2	0.27435157
40	1	2	0.27382390
148	1	2	0.27357396
441	1	2	0.27303057
477	1	2	0.27192895
512	1	2	0.26770091
197	1	2	0.26681023
136	1	2	0.26464568
141	1	2	0.26464201
169	1	2	0.26368934
85	1	2	0.26311249
674	1	2	0.26263389
572	1	2	0.26136293
574	1	2	0.26136293
576	1	2	0.26136293
103	1	2	0.26077174
699	1	2	0.26014754
538	1	2	0.25967508
792	1	2	0.25957209
429	1	2	0.25709638
248	1	2	0.25684373

663	1	2	0.25651857
560	1	2	0.25220784
453	1	2	0.25157511
212	1	2	0.25057796
614	1	2	0.24744204
184	1	2	0.24610437
666	1	2	0.24493682
557	1	2	0.24280753
432	1	2	0.24193049
547	1	2	0.24164273
61	1	2	0.24152837
796	1	2	0.24073649
603	1	2	0.23945706
521	1	2	0.23920759
527	1	2	0.23701876
533	1	2	0.23612939
534	1	2	0.23612939
535	1	2	0.23612939
536	1	2	0.23612939
537	1	2	0.23612939
525	1	2	0.23553672
147	1	2	0.23374380
742	1	2	0.23296610
721	1	2	0.23027190
514	1	2	0.22995266
253	1	2	0.22828324
135	1	2	0.22777346
306	1	2	0.22664070
112	1	2	0.22648699
583	1	2	0.22639799
78	1	2	0.22517895
401	1	2	0.22437680
505	1	2	0.22368587
435	1	2	0.22246518
44	1	2	0.22161969
520	1	2	0.22084522
32	1	2	0.22029359
383	1	2	0.21756751
737	1	2	0.21649039
768	1	2	0.21617331
233	1	2	0.21580881
522	1	2	0.21556666
355	1	2	0.21455803
598	1	2	0.21401851
479	1	2	0.21335519
740	1	2	0.21275400
102	1	2	0.21193044

751	1	2	0.21123078
37	1	2	0.20955107
772	1	2	0.20862391
154	1	2	0.20814022
72	1	2	0.20805443
523	1	2	0.20574314
671	1	2	0.20537453
133	1	2	0.20469693
467	1	2	0.20467732
153	1	2	0.20085392
68	1	2	0.20025247
137	1	2	0.19647833
517	1	2	0.19495832
80	1	2	0.19449504
431	1	2	0.19346416
690	1	2	0.19267086
764	1	2	0.19148533
196	1	2	0.19124523
430	1	2	0.19058859
478	1	2	0.18704440
513	1	2	0.18492188
236	1	2	0.18474432
732	1	2	0.18188811
693	1	2	0.18136842
581	1	2	0.17890353
75	1	2	0.17823144
511	1	2	0.17796946
143	1	2	0.17634858
229	1	2	0.17428978
771	1	2	0.17118858
417	1	2	0.17084856
202	1	2	0.17002161
687	1	2	0.16983831
403	1	2	0.16978361
71	1	2	0.16938677
364	1	2	0.16894408
503	1	2	0.16717173
299	1	2	0.16668626
339	1	2	0.16656569
531	1	2	0.16594615
623	1	2	0.16484053
643	1	2	0.16108053
88	1	2	0.15983308
334	1	2	0.15926345
770	1	2	0.15753228
744	1	2	0.15715757
606	1	2	0.15624364

404	1	2	0.15545663
658	1	2	0.15542555
51	1	2	0.15426206
590	1	2	0.15395611
472	1	2	0.15197879
501	1	2	0.15167840
232	1	2	0.15094812
243	1	2	0.15068402
381	1	2	0.15022922
193	1	2	0.14988947
679	1	2	0.14763271
766	1	2	0.14716541
393	1	2	0.14679902
610	1	2	0.14664595
320	1	2	0.14431557
530	1	2	0.14402201
349	1	2	0.14390056
146	1	2	0.14319193
760	1	2	0.14277459
568	1	2	0.14222650
444	1	2	0.13816380
119	1	2	0.13734860
254	1	2	0.13661648
524	1	2	0.13638058
375	1	2	0.13432680
786	1	2	0.13368754
99	1	2	0.13324666
515	1	2	0.13299799
608	1	2	0.13084438
455	1	2	0.12949632
415	1	2	0.12925907
397	1	2	0.12782983
548	1	2	0.12782983
416	1	2	0.12738249
377	1	2	0.12698187
785	1	2	0.12604881
225	1	2	0.12425574
592	1	2	0.12338019
516	1	2	0.12203345
648	1	2	0.12157918
198	1	2	0.12077089
635	1	2	0.12057574
762	1	2	0.11844705
668	1	2	0.11783041
694	1	2	0.11761809
617	1	2	0.11631445
756	1	2	0.11528061

23	1	2	0.11527691
93	1	2	0.11333426
595	1	2	0.11299618
611	1	2	0.11271484
139	1	2	0.11184990
491	1	2	0.11045485
333	1	2	0.10924221
683	1	2	0.10878345
586	1	2	0.10873941
252	1	2	0.10872235
369	1	2	0.10763246
97	1	2	0.10622165
787	1	2	0.10600980
366	1	2	0.10579067
387	1	2	0.10550844
778	1	2	0.10502318
758	1	2	0.10439167
296	1	2	0.10297291
151	1	2	0.10289671
108	1	2	0.10191068
20	1	2	0.10034112
311	1	2	0.09980592
486	1	2	0.09889007
676	1	2	0.09852672
685	1	2	0.09695004
213	1	2	0.09603089
638	1	2	0.09574213
627	1	2	0.09565087
110	1	2	0.09447614
90	1	2	0.09370254
745	1	2	0.09337922
677	1	2	0.09064148
368	1	2	0.08980785
476	1	2	0.08920436
651	1	2	0.08805401
618	1	2	0.08759896
379	1	2	0.08696192
484	1	2	0.08577537
83	1	2	0.08490854
570	1	2	0.08330657
63	1	2	0.08311078
780	1	2	0.07912507
354	1	2	0.07824568
42	1	2	0.07795480
788	1	2	0.07764139
156	1	2	0.07618639
631	1	2	0.07482723

```
 [ reached 'max' / getOption("max.print") -- omitted 467 rows ]
```

3d(KMeans clustering scatter plot)
Rcode:

```
# To plot the model for KMeans
> km.model <- eclust(scaled_data, FUNcluster = "kmeans", k=2, nstart=50)
```

3d (The ggplot2 of Defense vs. Sp. Def)
Rcode:

```
> # Extract Defense and Sp.Def columns from scaled_data
> def_SpDef <- scaled_data[, c("Defense", "Sp. Def")]
>
> # Add the clusters to def_SpDef
> def_SpDef$cluster <- as.factor(km.model$cluster)
>
> # Plot the scatterplot of Defense vs. Sp.Def
> ggplot(def_SpDef, aes(x = Defense, y = Sp.Def, color = cluster)) +
+  geom_point()
```

3d (The ggplot2 of HP vs. Attack)
Rcode:

```
> # Extract HP and Attack columns from scaled_data
> hp_atk <- scaled_data[, c("HP", "Attack")]
>
> # Add the clusters to hp_atk
> hp_atk$cluster <- as.factor(km.model$cluster)
>
> # Plot the scatterplot of HP vs. Attack
> ggplot(hp_atk, aes(x = HP, y = Attack, color = cluster)) +
+  geom_point()
```

3d (The dendrogram for Hierarchical clustering using average linkage)
Rcode:

```
> # Plot the dendrogram using average linkage
> hc.avg <- hclust(dist(scaled_data), method = "average")
> plot(hc.avg)
```

3d (The other dendrogram for Hierarchical clustering using average linkage)
Rcode:

```
> # Extract the selected features
> pt <- t(Pokemon_subset)
> # Create the average linkage
> hc.average <- hclust(dist(pt),method = "average")
> # To the plot the average linkage dendrogram
> plot(hc.average)
```