Classification of Pokemon by Stats using K-Means and Prediction of HP using Linear and Polynomial Regression

Raymundo Romero Arenas, A00570654
[1]ITESM Campus Querétaro
e-mail: A00570654@tec.mx

***Abstract* – Regression is a task that consists on predicting a value in a dataset. It can be done using any function. While some of them may fit to a linear model, others may perform better on a nonlinear one such as a polynomial function. K-Means is a clustering task that groups data points in order to find patterns among them. In this paper, it is aimed to perform a two different regressions on a Pokemon dataset to predict the HP of a Pokemon, facilitating its setting values, and also run a K-Means algorithm to cluster their stats in order to find patterns about how they are tiered, as an overview for its competitive format, both experiments yielding insights about how Pokemon are designed from scratch.**

***Keywords* – KMeans, Pokemon, Linear Regression, Machine Learning**

NOMENCLATURE

HP	Health Points
ML	Machine Learning
MSE	Mean Square Error
R2	Coefficient of Determination

I. INTRODUCTION

Pokemon is one of the biggest and most profitable franchises in the world. Just in 2021, it had an estimated revenue of 105 billion dollars worldwide. And with such a huge and still growing community, it has branched into several media and entertainment forms. One of the most prominent ones has been its competitive format, know as Pokemon VGC (Video Game Championship), in which players all over the world battle in a doubles format (two Pokemon at once) with a team of six Pokemon, using only four, under a limited amount of time and with special limitations, to win great prizes throughout the year. Although the game looks simple at first sight, its mechanics give a high level of complexity in which players need to design and fine-tune their Pokemon team based on a certain playstyle or strategy. Any attribute in a Pokemon must be taken in account: Statistcs, movesets, abilities, objects, strengths, weaknesses and training [1].

As new games go on sale and new generations are revealed, Game Freak, Pokemon's development team, has to set and balance the statistics of the newer Pokemon to fit into its already large roster, and this process is usually done by hand by the programmers, making it time-consuming. Looking at the base stats for each Pokemon, we can see that there's a mindset followed by the developers to balance them. For example, if a Pokemon has a high Attack stat, it will have a low Special Attack stat, and viceversa; same for Defense and Special Defense. Speed is influenced by the four previous stats, as it defines the probability of a Pokemon to attack first. However, HP (Health Points) is not correlated to any other stat and rather it is set in a more arbitrary way that makes it seem appear more independent to the rest.

Also, how Pokemon are perceived and used by the general player is influenced by how distributed by their stats. Inside Pokemon's competitive format there are tier systems to rank Pokemon based on their perceived power and usage in competitive play that dictate which can be used in various metagames, in order to balance battling and avoid rigid team design with Pokemon too powerful. However, these rankings fluctuate as new Pokemon are introduced and new strategies and movesets are discovered and playtested across seasons, making harder to predict the viability of a Pokemon [2].

Regression is a type of predictive analysis in which it is examined whether one or more variables can predict a good outcome or dependent variable. These estimates are used to explain and model the relationship between two or more variables, and in the case of Machine Learning (ML), doing it while minimizing its error. This representation can be done with any function, but by far the most commonly used is Linear Regression, which fits the data in a line with the form y = mx + b or y = m1x1 + ... + mNxN, due to its simplicity and ease to understand. However, there are models that use polynomial functions (with a degree greater or equal than 2) to represent data that may have a non-linear pattern. [3] [4].

There are several techniques to perform a regression model. Traditional ones such as matrix operations or basic arithmetic are very fast but restricted due to their implementation limitations, while those based on ML are slower but more flexible and error-free due to their optimization focus. One of the most known ones is Gradient Descent (GD), an optimization algorithm that finds the local minimum or maximum of a function while iteratively minimizing its error. It is easy to understand and implement, but also very powerful to model large quantities of data. [5]

K-Means is an unsupervised machine learning algorithm that groups similar data points together and finds underlying patterns by looking for a fixed number (k) of clusters in a dataset. The process begins by randomly selecting a first group of centroids, which are used as the beginning points for every cluster, and then iterative calculations are done to optimize their positions until they are stable. Although it is easy to understand and implement, it doesn't has the best performance as other clustering techniques because slight variations in the data could lead to high variance [6] [7].

In this paper, we will perform two tasks using Pokemon: First, we will predict the HP of a Pokemon based on its stats (Attack, Defense, Special Attack, Special Defense and Speed)

using Linear and Polynomial Regression in order to automate its setting process. Second, we will cluster the Pokemon into groups based on pairs of correlated stats using K-Means, in order to give a general overview into how Pokemon are ranked on competitive tier systems. This case will yield insights into Pokemon mechanics, and it may serve as an inspiration to expand analysis to unexplored attributes of Pokemon such as moves, abilities, objects, between others.

The dataset used in these implementations was found on Kaggle, and it contains the information of all Pokemon up to the 6th generation, which amount to 729 in total. 8 columns were extracted from the dataset, which document the name and stats values of each Pokemon.

II. IMPLEMENTATION

A. Linear Regression

1) Data loading: Using Pandas library from Python, we will download our Pokemon dataset in a .CSV format and define our attributes and predictors. In this case, the HP stat will be our predictor, while the other stats will be our attributes (Attack, Defense, Special Attack, Special Defense, Speed).

2) Data preprocessing: To perform the regression, we need to set first the Learning Rate (α) in which the model will calculate the parameters per epoch. In this case, we will set it to 0.01 to avoid divergence. Then, we define the initial values of the parameters. Since we have five attributes, we will have six parameters (N+1) to do the one-to-one mapping, leaving the first parameter as the one to map to the bias.

After that, we will process the data by inserting it into Python lists of lists, one for the attributes and another one for the predictors. With our data ready, we will split it into a training and test set to teach our model and proceed with the Gradient Descent pre-processing.

3) Bias: As described previously, the first parameter in our list of parameters will be mapped to the bias (b) in our linear equation. However, our data doesn't has an attribute that represents it. In order to account for it, we will manually add it by inserting a 1 at the beginning of each of our examples.

4) Scaling: Although the difference between the dataset values is not considerable (0-250), the calculations may be too big for Python to do. Therefore, we will scale our training set through Standarization Scaling, which will normally distribute the values by using the following equation:

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

where,
- x = Example attribute value
- μ = Mean of the attribute values
- σ = Standard Deviation of the attribute values

5) Hypothesis: Before proceeding to the Gradient Descent method, we have to talk about how the y-hypothesis is calculated. This is done by substituting the attribute values of an example into the linear equation and multiplying them with the parameter values of the current epoch.

6) Mean Square Error: Mean Square Error (MSE) is the average squared difference between the estimated values and the actual value of a data point. In Gradient Descent, we find the optimal parameters for our linear regression by minimizing it per epoch. It is calculated with the following formula:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_{hyp} - y_{real})^2 \tag{2}$$

where,
- N = Number of examples
- y_{hyp} = Hypothesis Value
- y_{real} = Real Value

Hypothesis Value is the result of performing the hypothesis calculation described in the last subsection on an example, while Real Value is the predictor value of the example.

7) Gradient Descent: To perform the Linear Regression, first, we define the number of epochs we want our model to run and create a list in which we will store the MSE of each epoch. Then, we iterate through each parameter to calculate its new value using the Gradient Descent equation:

$$m_j = m_j - \frac{\alpha}{N}\sum_{i=1}^{N}(yi_{hyp} - yi_{real})x_{ij} \tag{3}$$

where,
- α = Learning Rate
- N = Number of examples
- y_{hyp} = Hypothesis Value
- y_{real} = Real Value
- x_{ij} = Parameter Attribute Value

Parameter Attribute Value is the value of the attribute whose parameter we are currently calculating.

After getting the new values for the parameters, we can calculate the MSE of the new equation and insert it into the Epoch MSE list. Once the parameters don't change between epochs, or we reach our epoch limit, we can state them as the final parameters of our linear regression. Finally, we plot the Epoch MSE list and check if the MSE decreases on each epoch until it arrives to a stable state. Although it may be large (depending on the dataset), it can be trusted if we have a high coefficient of determination.

8) Coefficient of Determination: To calculate the coefficient of determination (R2) of our model, first we need to get the predicted values of the examples from our test set. Therefore, we extract the optimal parameters from the Gradient Descent method and multiply them to the attributes values of the test examples to get them. Once finished, we can calculate the R2 with the following formula:

$$R2 = 1 - \frac{SSR}{SST} \tag{4}$$

where,
- SSR = Sum of Square Residuals
- SST = Sum of Square Totals

Sum of Square Residuals (SSR) is the sum of squares of the difference between the Predictor Real Value and Predictors Mean Value, while Sum of Square Totals (SST)

is the sum of squares of the difference between the Predictor Predicted/Hypothesis Value and Predictors Mean Value. The more R2 is closer to 1, the better our model is able to predict a value, while the closer to 0, the better the model is to calculate the mean of the predictor.

B. Polynomial Regression

Sometimes the data may have relationships between attributes and predictors that can't be totally represented with a straight line. In those cases, rather than using Linear Regression to fit our data, we can instead use Polynomial Regression to fit it into a curvature that better suits our predictions. To do so, we need to understand to adapt our data into a polynomial function. However, for this case we will consider the following:

1. This implementation was done using scikit-learn, a Python framework for Machine Learning, contrary to the Linear Regression, which was done from scratch
2. This implementation was done with single and multiple attributes: The latter to compare its results with those from our Linear Regression from scratch, and the former to visualize our data into a plot. The code differences between both forms will be stated along the steps.

1) Data loading: Using Pandas library from Python, we will download our Pokemon dataset in a .CSV format and define our attributes and predictors. In this case, the HP stat will be our predictor.

- For Single Polynomial Regression, any stat can be our attribute (in the implementation it can be chosen which one to use).
- For Multiple Polynomial Regression, the other stats will be our attributes (Attack, Defense, Special Attack, Special Defense, Speed).

2) Degree: To perform the regression, we need to set first the degree of our polynomial function. Scikit-learn generates a new feature matrix with all polynomial combinations of the features with degree less than or equal to the specified one.

- For Single Polynomial Regression, as we only have one attribute (a), it only generates all polynomial combinations of a.
  - Degree 2 = (1, a, a^2)
  - Degree 3 = (1, a, a^2, a^3)
- For Multiple Polynomial Regression, as we have five attributes in this case (a, b, c, d, e), it will generate all the polynomial combinations of them.
  - Degree 2 = (1, a, b, c, d, e, a^2, ab, ac, ad, ae, b^2, bc, bd, be, c^2, cd, ce, d^2, de, e^2)
  - Degree 3 = (1, a, b, c, d, e, a^2, ab, ac, ad, ae, b^2, bc, bd, be, c^2, cd, ce, d^2, de, e^2, a^3, a^2b, a^2c, a^2d, a^2e, ab^2, abc, abd, abe, ac^2, acd, ace, ad^2, ade, ae^2, b^3, b^2c, b^2d, b^2e, bc^2, bcd, bce, bd^2, bde, be^2, c^3, c^2d, c^2e, cd^2, cde, ce^2, d^3, d^2e, de^2, e^3)

3) Data transformation: After declaring the degree of our polynomial regression, we proceed to transform the data by fitting it into the polynomial form. Once ready, we will split it into a training and test set to teach our model and proceed with the regression.

4) Polynomial Regression: For the Polynomial Regression, we call the Linear Regression function from scikit and fit the training data in it. With that we can predict the values of our test set. In order to check the efficiency of our model, we can calculate its MSE and R2 with their scikit functions.

5) Plotting (Single Polynomial Regression only): In order to visualize our regression, we integrate the test set and predictions into a Pandas dataframe, sort it by the attribute, and display the real and predicted values through a scatter plot. This can be done with any of the stats against the HP.

C. K-Means

1) Data loading: Using Pandas library from Python, we will download our Pokemon dataset in a .CSV format and define our pair of attributes to cluster. To plot our data, we will use Seaborn, a Python data visualization library based on Matplotlib to ´draw statistical graphics.

2) Within-Cluster Sum of Squares (WCSS): Within-Cluster Sum of Squares (WCSS) is the sum of squares of the distances of each data point in all clusters to their respective centroids. The idea is to minimize the sum while maximizing the number of clusters needed. However, we may reach to a point where all data points serve as a centroid and therefore, the WCSS is equal to 0. To find the optimum number for clusters (value for K), we can use the Elbow Method, in which we vary the number of clusters from 1 to 10 and do the K-Means to calculate and plot their WCSS. The resulting graph will have an elbow shape as the WCSS starts to decrease while the number of clusters increases, so we select K based on the rate of WCSS decrease. Once we have the optimal K-value, we can redo the algorithm to plot the clustered data.

3) Implementation: To perform the K-Means algorithm, first we create a list in which we will store the index of the centroid assigned to each example. Then, we select K random examples from the dataset and extract their values into a list of lists to use them as our initial centroids coordinates. While searching for the optimal centroids, we start each iteration assuming that the mean distance from all data points to them is infinite. We calculate the distance of each example to the current evaluated centroid through Euclidean Distance. If an example has a smaller distance than the mean, we update the mean distance and assign to it the ID of the centroid. Once all examples are evaluated, we group them by their current cluster, and calculate the new centroids coordinates thorugh their mean. If the new centroids are the same as those from the previous ones, we return the list of centroids and the clustered data, otherwise, we update the centroids and perform another iteration to keep searching for the optimal coordinates.

III. RESULTS

A. Linear Regression

After running our Linear Regression with various epochs, we get the following results:

Epochs	50	100	1000	10000
Bias	7.0169	7.1353	7.2216	6.0826
m_0	4.2881	4.6578	7.9966	9.7069
m_1	3.7016	2.4746	-6.7256	-10.1241
m_2	6.8546	6.9663	5.6047	3.5381
m_3	10.1656	10.3389	14.0567	18.5405
m_4	15.6859	15.9602	16.2958	15.4737
MSE	652.3975	645.2068	614.4745	610.6454
R2	0.999862	0.999857	0.999811	0.999798

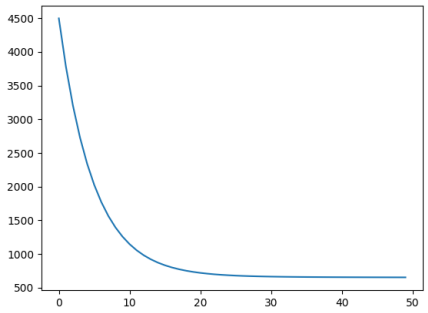

Fig. 1. MSE Plot (50 Epochs)

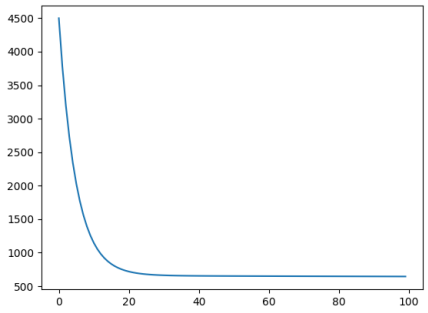

Fig. 2. MSE Plot (100 Epochs)

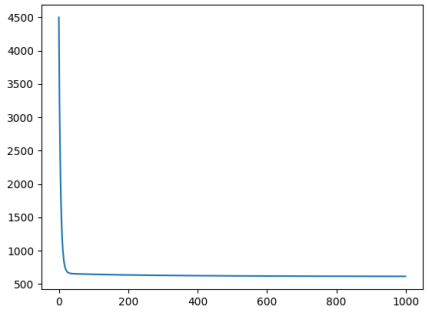

Fig. 3. MSE Plot (1000 Epochs)

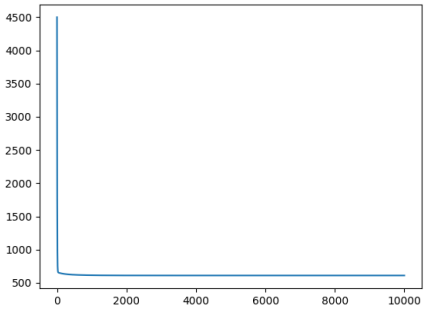

Fig. 4. MSE Plot (10000 Epochs)

Based on the results, we can see that our model will always give an MSE higher than 600. Although our HP predictions for a Pokemon will be very off from normal values found of the dataset, the R2 indicates that it is precise. Thus, our model may be underfitting, and therefore it requires more attributes to give more accurate predictions.

B. Single Polynomial Regression

After running our Single Polynomial Regression with each Pokemon stat (Attack, Defense, Sp.Attack, Sp.Defense, Speed) to predict HP, we get the following results:

Degree 2		
	Attack	Defense
a	0.3645	0.6262
a^2	0.0002	-0.0023
MSE	21.2802	22.0316
R2	0.2170	0.1608

Degree 2			
	Sp. Attack	Sp. Defense	Speed
a	0.6805	1.0900	0.3461
a^2	-0.0022	-0.0044	-0.0013
MSE	21.8973	21.4198	23.5700
R2	0.1710	0.2068	0.0395

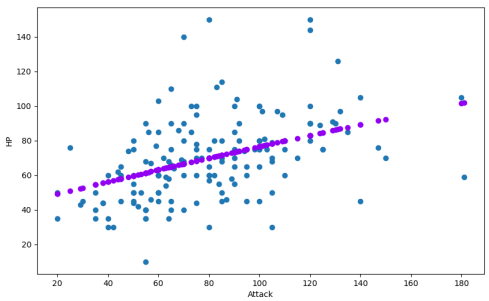

Fig. 5. Prediction Plot (Attack vs. HP)

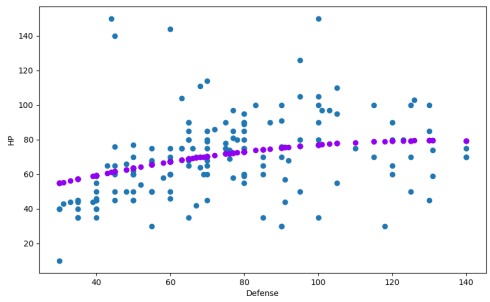

Fig. 6. Prediction Plot (Defense vs. HP)

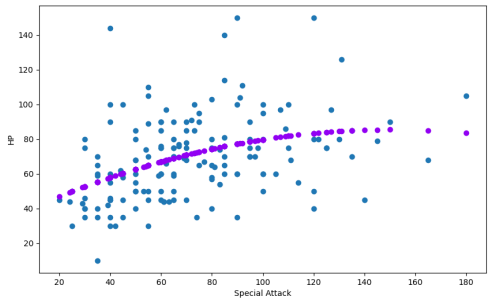

Fig. 7. Prediction Plot (Special Attack vs. HP)

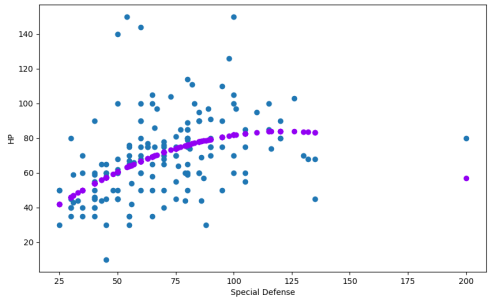

Fig. 8. Prediction Plot (Special Defense vs. HP)

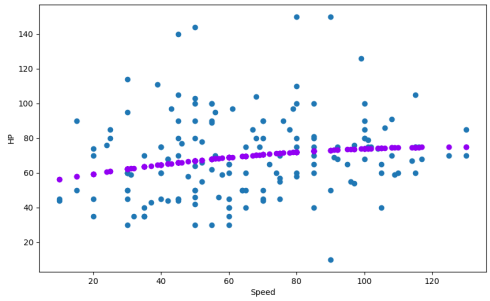

Fig. 9. Prediction Plot (Speed vs. HP)

Using a 2° degree polynomial, we get a model with a low MSE and R2 score on each Pokemon stat, meaning that the predictions are highly correct but tend to lean more towards the stat mean. In contrast to the Linear Regression, it is more accurate but has a lower correlation between variables.

Degree 3		
	Attack	Defense
a	-1.2015	0.1002
a^2	$1.982E-02$	$3.310E-03$
a^3	$-7.459E-05$	$-1.718E-05$
MSE	21.2682	22.3284
R2	0.2179	0.1380

Degree 3			
	Sp. Attack	Sp. Defense	Speed
a	0.5672	0.8774	-0.1448
a^2	$8.278E-04$	$-2.068E-03$	$5.597E-03$
a^3	$-5.247E-06$	$-7.769E-06$	$-2.866E-05$
MSE	21.9259	21.4821	23.6072
R2	0.1688	0.2021	0.0364

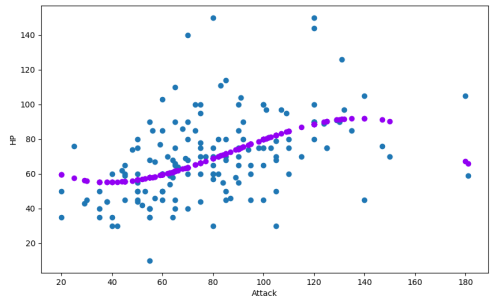

Fig. 10. Prediction Plot (Attack vs. HP)

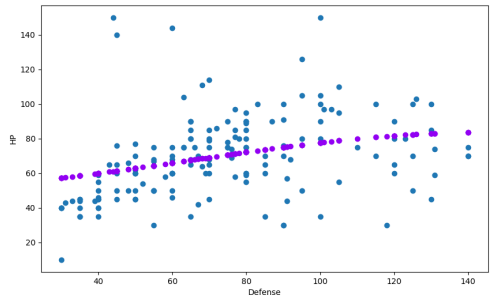

Fig. 11. Prediction Plot (Defense vs. HP)

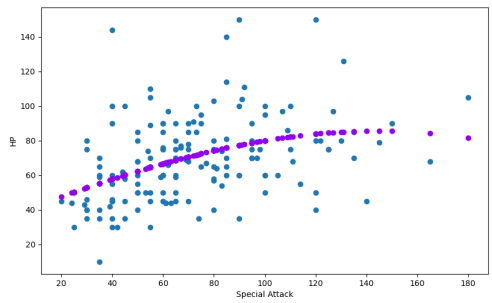

Fig. 12. Prediction Plot (Special Attack vs. HP)

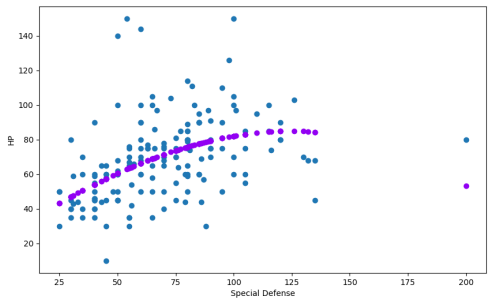

Fig. 13. Prediction Plot (Special Defense vs. HP)

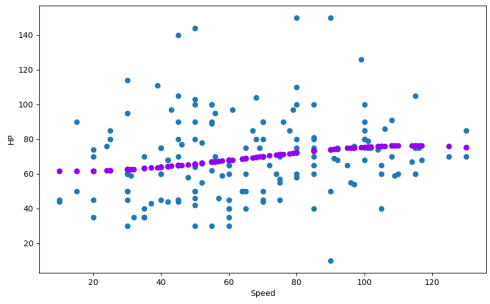

Fig. 14. Prediction Plot (Speed vs. HP)

Using a 3° degree polynomial, we get a model with a slightly lower MSE and R2 score than a 2° degree polynomial on each Pokemon stat. However, the change is so minimum that it can be implied that both can be interchangeably used and provide almost similar results. Therefore, we can say that a 2° degree polynomial provides an optimal model to predict the HP of a Pokemon with a low error rate.

C. Multiple Polynomial Regression

After running our Multiple Polynomial Regression, we get the following results:

Degree 1			
m_0	0.3206	m_3	0.2928
m_1	-0.1208	m_4	-0.1149
m_2	0.1231		
MSE	18.6962	R2	0.327782

Degree 2			
m_0	0.7103	m_1^2	$1.693E-03$
m_1	-0.7615	m_1m_2	$3.557E-03$
m_2	0.0155	m_1m_3	$-7.614E-03$
m_3	0.1150	m_1m_4	$7.944E-03$
m_4	-0.4935	m_2^2	$-4.865E-04$
m_0^2	$-1.333E-03$	m_2m_3	$-2.526E-03$
m_0m_1	$2.833E-03$	m_2m_4	$6.898E-04$
m_0m_2	$7.526E-04$	m_3^2	$3.806E-03$
m_0m_3	$-6.112E-03$	m_3m_4	$-3.230E-04$
m_0m_4	$1.854E-04$	m_4^2	$-2.458E-04$
MSE	21.2659	R2	0.1303

Degree 3			
m_0	-0.2339	$m_0m_1m_4$	$-8.002E-07$
m_1	-1.1960	$m_0m_2^2$	$-1.985E-05$
m_2	-0.4815	$m_0m_2m_3$	$1.221E-05$
m_3	1.1675	$m_0m_2m_4$	$5.953E-05$
m_4	-1.1545	$m_0m_3^2$	$-1.686E-05$
m_0^2	$3.858E-03$	$m_0m_3m_4$	$-3.079E-05$
m_0m_1	$8.835E-03$	$m_0m_4^2$	$1.923E-05$
m_0m_2	$1.759E-02$	m_1^3	$-1.962E-05$
m_0m_3	$-7.923E-03$	$m_1^2m_2$	$2.852E-05$
m_0m_4	$-4.926E-03$	$m_1^2m_3$	$1.407E-04$
m_1^2	$5.765E-03$	$m_1^2m_4$	$-1.251E-04$
m_1m_2	$-3.103E-03$	$m_1m_2^2$	$-9.828E-05$
m_1m_3	$-1.801E-02$	$m_1m_2m_3$	$1.389E-04$
m_1m_4	$3.487E-02$	$m_1m_2m_4$	$7.077E-05$
m_2^2	$2.654E-03$	$m_1m_3^2$	$-2.153E-04$
m_2m_3	$-1.024E-03$	$m_1m_3m_4$	$2.133E-04$
m_2m_4	$4.709E-04$	$m_1m_4^2$	$-2.011E-04$
m_3^2	$1.110E-02$	m_2^3	$-1.446E-06$
m_3m_4	$-2.073E-02$	$m_2^2m_3$	$9.288E-05$
m_4^2	$8.417E-03$	$m_2^2m_4$	$-2.499E-05$
m_0^3	$-1.120E-05$	$m_2m_3^2$	$-1.325E-04$
$m_0^2m_1$	$1.812E-05$	$m_2m_3m_4$	$-1.983E-05$
$m_0^2m_2$	$-8.599E-05$	$m_2m_4^2$	$-3.383E-04$
$m_0^2m_3$	$3.535E-04$	m_3^3	$9.648E-05$
$m_0^2m_4$	$-3.814E-06$	$m_3^2m_4$	$-9.537E-05$
$m_0m_1^2$	$-5.292E-05$	$m_3m_4^2$	$1.486E-04$
$m_0m_1m_2$	$-5.241E-05$	m_4^3	$-1.864E-05$
$m_0m_1m_3$	$1.966E-05$		
MSE	21.0765	R2	0.1457

Comparing the 1°, 2° and 3° degree polynomial models, we can see that the first one provides the best results in terms of MSE and R2 score, while the latters have minimal differences between them. In either case, this means that the predictions are highly correct but tend to lean more towards the stats mean. However, under the strict definition of polynomial function, we would have to discard the linear model, therefore the 2° degree polynomial would be the optimal one to predict HP given all Pokemon stats.

Finally, if we compare both Single and Multiple Polynomial Regressions, we see that the MSE and R2 scores have similar values, being less than 25 and less than 0.2 respectively. This would imply that any of them can predict confidently the HP of a Pokemon regardless of the stat used.

D. K-Means

After running our K-Means algorithm with various pairs of stats from our Pokemon dataset, we get their Elbow Point Graphs with the WCSS plotted from 1 to 10 clusters (see Figures 1-9). By analyzing them, we can see that the optimal K-value across all of them is 4, since it has the latest non-linear rate of WCSS decrease. Therefore, we will cluster our Pokemon into four groups on all of our cases.

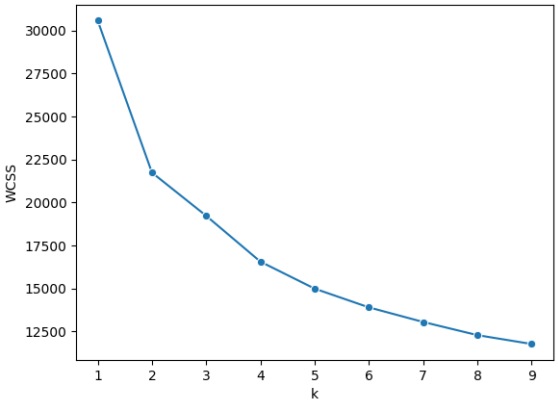

Fig. 15. Elbow Point Graph (Attack vs. Defense)

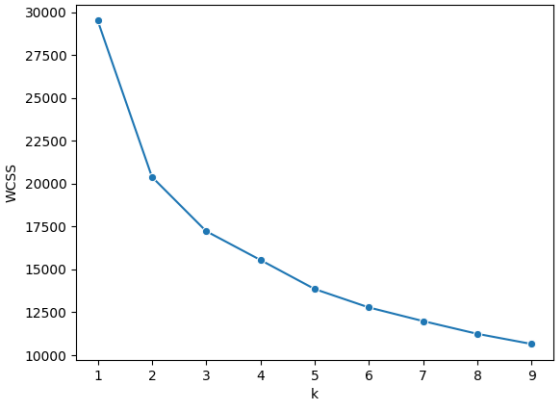

Fig. 16. Elbow Point Graph (Sp. Attack vs. Sp.Defense)

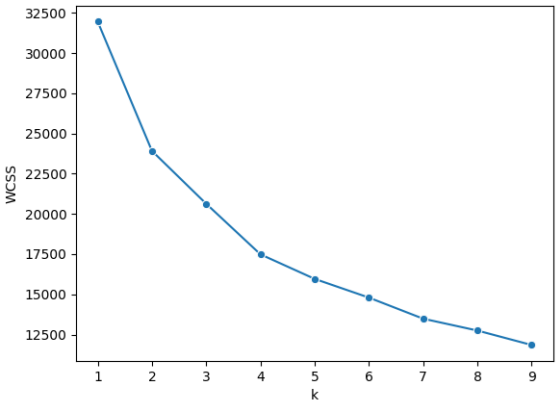

Fig. 17. Elbow Point Graph (Attack vs. Sp.Attack)

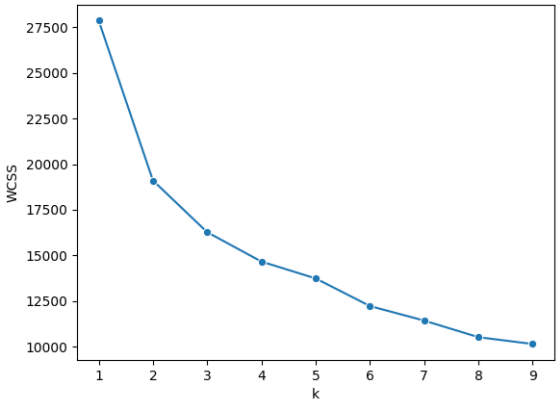

Fig. 18. Elbow Point Graph (Defense vs. Sp.Defense)

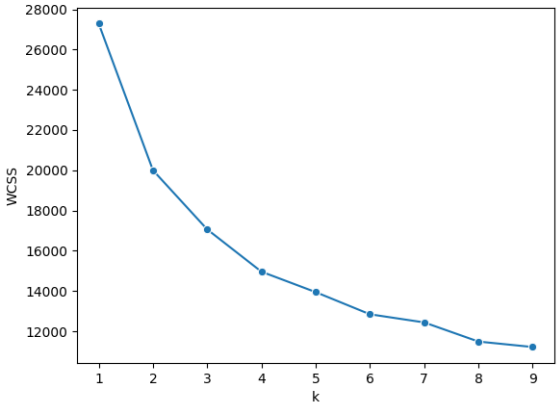

Fig. 19. Elbow Point Graph (HP vs. Speed)

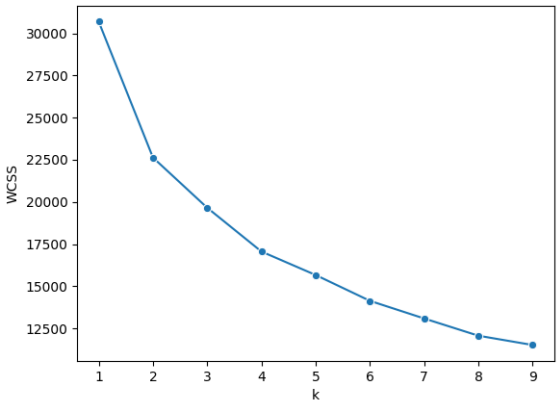

Fig. 20. Elbow Point Graph (Speed vs. Attack)

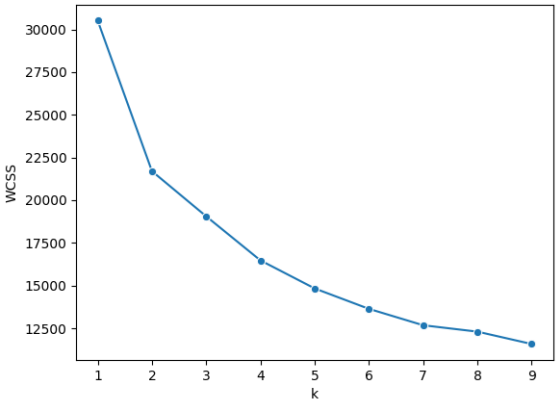

Fig. 21. Elbow Point Graph (Speed vs. Sp.Attack)

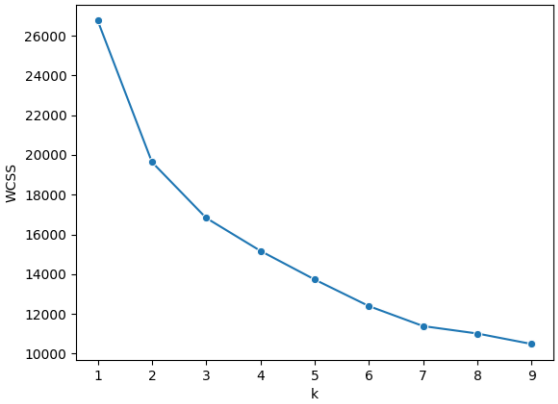

Fig. 22. Elbow Point Graph (HP vs. Defense)

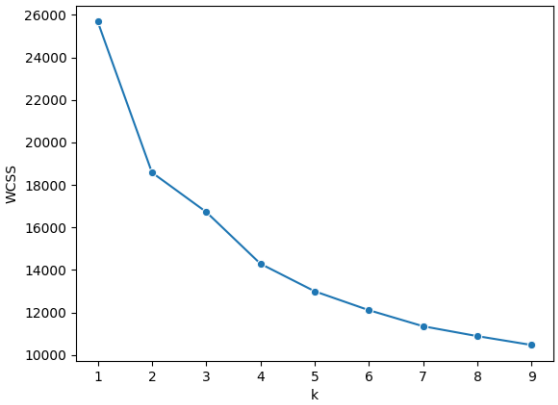

Fig. 23. Elbow Point Graph (HP vs. Sp.Defense)

After defining the number of centroids to 4, we cluster the Pokemon get the following plots (see Figures 10-18). We can see that the Pokemon fall into four categories on each case:

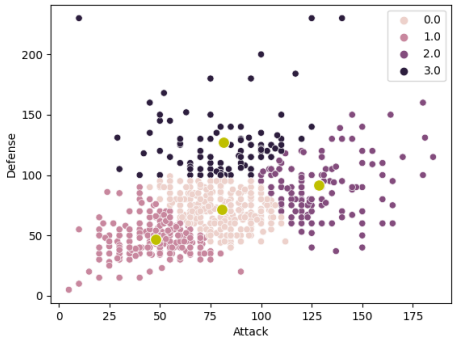

Fig. 24. Cluster Scatter plot (Attack vs. Defense)

1. Attack < 70 and Defense < 70
2. Attack (70 < X < 100) and Defense (70 < Y < 100)
3. Defense > 100
4. Attack > 100

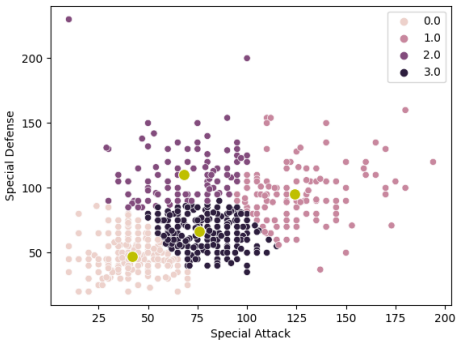

Fig. 25. Cluster Scatter plot (Sp.Attack vs. Sp.Defense)

1. Sp.Attack < 70 and Sp.Defense < 70
2. Sp.Attack (70 < X < 100) and Sp.Defense (70 < Y < 100)
3. Sp.Defense > 100
4. Sp.Attack > 100

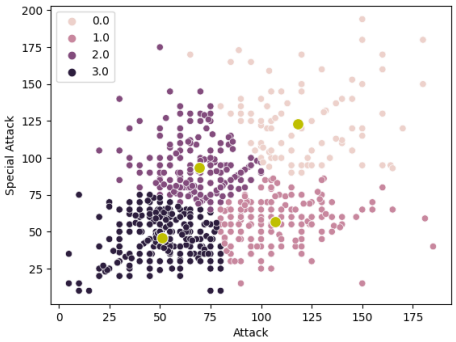

Fig. 26. Cluster Scatter plot (Attack vs. Sp. Attack)

1. Attack < 75 and Sp.Attack < 75
2. Attack > 75 and Sp.Attack < 85
3. Attack < 85 and Sp.Attack > 75
4. Attack > 85 and Sp.Attack > 85

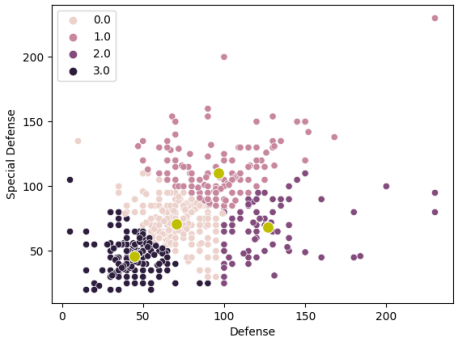

Fig. 27. Cluster Scatter plot (Defense vs. Sp. Defense)

1. Defense < 70 and Sp.Defense < 70
2. Defense (70 < X < 100) and Sp.Defense (70 < Y < 100)
3. Defense > 100
4. Sp.Defense > 100

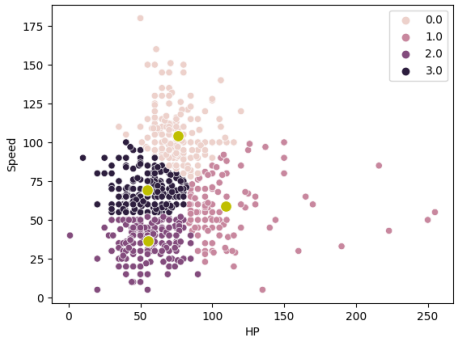

Fig. 28. Cluster Scatter plot (HP vs. Speed)

1. HP < 80 and Speed < 50
2. HP < 80 and Speed (50 < Y < 100)
3. HP > 80 and Speed > 100
4. Speed > 100

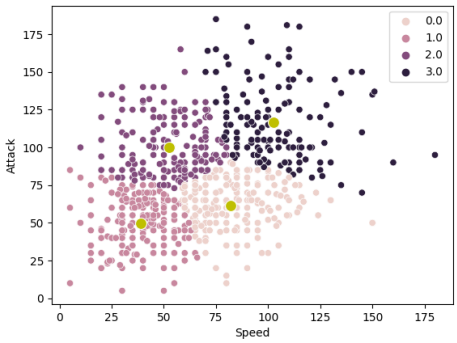

Fig. 29. Cluster Scatter plot (Speed vs. Attack)

1. Speed < 65 and Attack < 75
2. Speed > 65 and Attack < 85
3. Speed < 75 and Attack > 75
4. Speed > 75 and Attack > 85

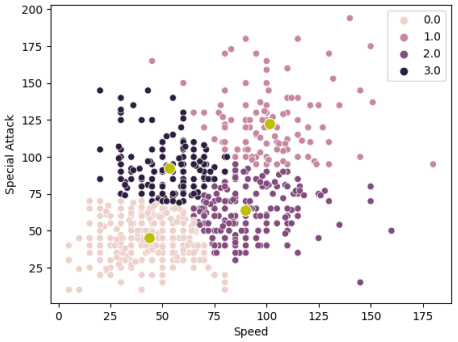

Fig. 30. Cluster Scatter plot (Speed vs. Special Attack)

1. Speed < 65 and Sp.Attack < 75
2. Speed > 65 and Sp.Attack < 85
3. Speed < 75 and Sp.Attack > 75
4. Speed > 75 and Sp.Attack > 85

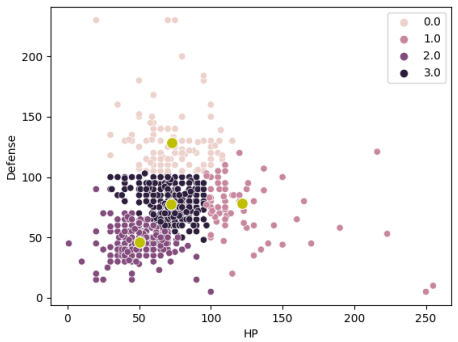

Fig. 31. Cluster Scatter plot (HP vs. Defense)

1. HP < 70 and Defense < 70
2. HP (70 < X < 100) and Defense (70 < Y < 100)
3. HP > 100 and Defense < 100
4. HP < 100 and Defense > 100

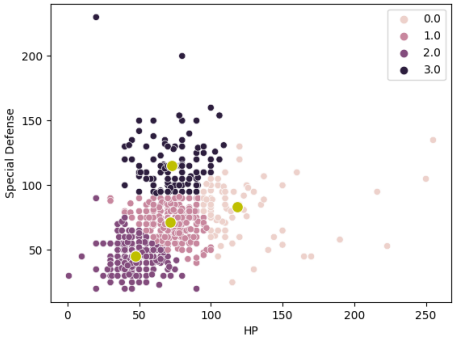

Fig. 32. Cluster Scatter plot (HP vs. Special Defense)

1. HP < 70 and Sp.Defense < 70
2. HP (70 < X < 100) and Sp.Defense (70 < Y < 100)
3. HP > 100 and Sp.Defense < 100
4. HP < 100 and Sp.Defense > 100

If we analyze the plots, we can see that the algorithm tends to cluster the stats into three kinds of patterns:

Fig. 33. K-Means Pattern 1

Fig. 34. K-Means Pattern 2

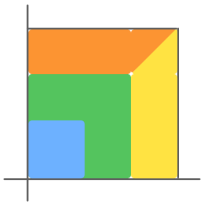

Fig. 35. K-Means Pattern 3

1. Pattern 1
    - Attack vs. Defense
    - Sp.Attack vs. Sp.Defense
2. Pattern 2
    - Attack vs. Sp.Attack
    - Speed vs. Attack
    - Speed vs. Sp.Attack
3. Pattern 3
    - Defense vs. Sp.Defense
    - HP vs. Defense
    - HP vs. Sp.Defense

Based on this clustering patterns, we can get some interpretations about how Pokemon stats are balanced out:

1. Based on Defense and Sp.Defense, Pokemon are always clustered within the ranges [0,70], [70,100], [100,250], and it is proportionally related to HP. However, some Pokemon tend to lean more towards one of them.
2. Based on Attack and Sp.Attack, Pokemon are always clustered within the ranges [0,65], [65,85], [85,250], and it is related to Speed. However, in any of these stats, one of them will always be slightly higher on a Pokemon. Cases in which they are both low or high proportionally are less common.
3. Based on Attacking and Defensive stats, Pokemon are always clustered within the ranges [0,70], [70,100], [100,250]. Pokemon whose stats are less than 100 tend to be divided into 2 groups, depending on which stat is slightly higher, while those with one stat higher than 100 tend to favor it rather than balance it.
4. While rerunning the algorithm with 3 clusters, we found that the mid-groups from Pattern 2 are merged into one. This may be because the algorithm bases the centroid coordinates on their numeric values and not on their position. In Pattern 3, the mid-group is split and shared between the low and high tier ones, while in Pattern 1, the mid-group extends to the data outliers (i.e. Pokemon that have one stat extermely high and other extremely low).

Although this experiment helps to get a general idea into how Pokemon stats are distributed and related between them, it doesn't give us a full picture of the quality or competitive viability of a Pokemon, since it can only take into account 2 out of 6 stats per case. In order to have a definitive result into how Pokemon are tiered, we would have to overlap the plots and find related areas between them. However, we can reach so far two main conclusions by looking at the clusters:

1. Low-tier Pokemon have base stats lower than 70
2. High-tier Pokemon have base stats higher than 100
3. Most Pokemon have their base stats between 70-100

IV. CONCLUSIONS

From the tests performed to our Pokemon dataset through Regression and K-Means algorithms, we can summarize the learnings into these points:

1. A Linear Regression done from scratch provides a model with a R2 score higher than 95% but a MSE higher than 600 after 10000 epochs, indicating a confident regression task but with predicting values very off from the expected HP values for a Pokemon.
2. A Polynomial Regression using scikit (either single or multiple) provides a model with a MSE lower than 25 but a R2 score less than 33% on all cases, indicating an accurate prediction task but a regression task whose confidence can't be fully validated.
3. Performing a K-Means clustering into the Pokemon stats shows that they are grouped into three clusters, with ranges of [0,70], [70,100], [100,250], with one of them divided into two subgroups, each one representing a stat preference of a Pokemon. It applies to all pairs of related stats, implying a baseline connection when setting their values into a Pokemon.

Further analysis on regression should focus on performing these methods on Pokemon stats at a competitive level (Lv.50) or max level (Lv.100), or by considering additional variables such as evolution line and nature. Regarding K-Means, there are opportunity areas to cluster the Pokemon with all their stats considered at once, as well as by performing the algorithm into other aspect of Pokemon such as moves and types.

REFERENCES

[1] V. Road, "¿Qué es pokémon VGC? - introducción al formato de competición oficial", *Victory Road*, Jul 2021, URL: https://victoryroadvgc.com/es/pokemon-vgc/.

[2] S. University, "An introduction to Smogon's tier system", URL: https://www.smogon.com/sm/articles/sm_tiers.

[3] S. Solutions, "What is linear regression?", , Aug 2021, URL: https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/what-is-linear-regression/.

[4] J. Brownlee, "Linear regression for machine learning", , Aug 2020, URL: https://machinelearningmastery.com/linear-regression-for-machine-learning/.

[5] R. Kwiatkowski, "Gradient descent algorithm, a deep dive", , May 2021, URL: https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21.

[6] D. M. J. Garbade, "Understanding K-means clustering in machine learning", , Sep 2018, URL: https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1.

[7] Khushijain, "K-means clustering: Python implementation from scratch", , Jul 2021, URL: https://medium.com/nerd-for-tech/k-means-python-implementation-from-scratch-8400f30b8e5c.