Course Number: CMPS 320 *Instructor: Professor Tweneboah*

Machine Learning in Pokemon

Maxwell Bilyk [1], Christopher Hakkenberg [2], Luke Ladas[3,], and Sikander Raja[4]

[1] Data Science Major; mbilyk@ramapo.edu
[2] Data Science Major; chakkenb@ramapo.edu
[3] Computer Science Major: sraja2@ramapo.edu
* Computer Science Major: lladas@ramapo.edu

Abstract:

In our project in regards to the pokemon dataset we wanted to test some things and see if they were true. Our Pokemon's Indicators are important measures for monitoring the health of the pokemon and identifying potential issues and metrics involving our pokemon. This research examines the catch rate ,is legendary, whether generation Attack and defense are correlated , pokemon type , and finally body style. All of the indicators above provide meaningful insights into the state of our pokemon in the dataset.Utilizing data collected by Alberto Rabbadas and uploaded to the internet site Kaggle. The data is analyzed to identify patterns and trends in the 6 research questions to accurately predict catch rate, finding clusters that in our pokemon, accurately predict the type of a pokemon, accurately predict if a pokemon is legendary and can we accurately predict pokemons body style. From the use of our machine learning algorithms we found out that as stats increase, catch rate decreases. As generation increases, so does the catch rate. When predicting a Pokemon's type and body style, there was little correlation between the features and a Pokemon's type. We can safely say that for our five features when predicting if a Pokemon is legendary, height, weight, catch rate, total stats, and gender are highly related to being legendary. When it comes to the attack and defense power while k means clustering there was no sort of pattern when analyzed. In addition to that during the dendrogram analysis there was no sort of pattern either.

Keywords: Deep learning, Scaling, Multiple linear regression,Select Vector Machine, Random Forest, Decision Tree, Cluster, Supervised model, Pandas, numpy, seaborn, matplotlib, Jupyter Notebook, Pokemon

1. Introduction

In recent years, the field of machine learning has experienced unprecedented growth and has found numerous applications in diverse domains, ranging from healthcare and finance to transportation and entertainment. One such domain that has garnered significant interest is the world of Pokemon. With its rich dataset of creatures, abilities, moves, and battle outcomes, Pokemon provides a captivating platform to study and apply machine learning algorithms. Pokémon is a multimedia franchise that began in 1996 with video games for the Game Boy. It features fictional creatures called Pokémon that trainers catch and train for battles. With over 800 species, players aim to become Pokémon Masters by collecting and battling with their teams. The franchise includes an animated TV series, Video games, movies, and merchandise. It has become a global phenomenon, capturing the hearts of millions of fans of all ages around the world. For the purpose of this presentation, we will be simulating as a team that works

in the part of the company that deals with the specific stats of the Pokémon. This research paper aims to delve into the fascinating intersection of machine learning and Pokemon, investigating the potential of various algorithms to analyze and model the vast troves of data within the Pokemon universe. We will explore how these algorithms can be leveraged to extract meaningful information, uncover hidden relationships, and enhance decision-making in different aspects of the Pokemon gameplay.

2. Materials and Methods

Overview of data - This dataset is focused on the stats and features of the Pokémon in the video games. The data was recorded by Alberto Rabbadas and published online. The dataset contains data on 721 Pokémon in total. There are 23 features in the dataset. For our main dataset the name wasn't necessary so it was dropped from the dataset. However for all of our own predictions we used different methods and features to predict. Dataset found: https://www.kaggle.com/datasets/alopez247/pokemon

2.1 Predicting Catch Rate

Since catch rate is a numerical target variable that exists within the dataframe, predicted it is a supervised regression problem. The materials used to solve it are python and jupyter notebook. The libraries used are pandas, numpy, seaborn, matplotlib, and multiple extensions of ski-kit learn. The data for this problem was normalized using the min-max scaling method in order to normalize the data. The multiple linear regression model used, uses the best subset method. The random forest regressor uses 1000 trees with a depth of 10 for each tree. The voting regressor uses hard voting in order to come up with the mean of all the regression methods combined.

2.2 Prediction of Pokemon Type

This question was dealt with as a supervised learning problem using classification of various techniques. Our predictor for classification was the 'Type_1' column in our dataset and the features that were chosen to predict if a Pokemon was a specific type was 'HP', 'Attack', 'Defense', 'SP_Atk', 'SP_Def', 'Speed', 'Generation', 'isLegendary', 'Height_m', 'Weight_kg', 'Catch Rate' 'Color', 'Body_Style' and 'has Gender'. We had to deal with many features being dropped for this question, for varying reasons. One for being irrelevant to our test, for example, certain features such as egg type essentially would give away what type our pokemon is since they share a lot of similar values. Probability of being male, hasMegaEvolution and 'Total' were removed because they were redundant or irrelevant so they ended up having either little impact to testing models or they actually harmed the models by lowering accuracies. Any pokemon that had a secondary type (any pokemon where 'Type_2' was not NaN) was dropped because it would make prediction very complex. Type 1 has 18 unique values and so does Type 2, meaning that to potentially guess what type a pokemon would be if it had a secondary type or not would make over 300 combinations of pokemon types to choose from. Variables that were categorical like body style and color were treated with one hot encoding and had dummy variable columns to make sure each value was weighted equally. This model was heavily skewed in some features (mainly positive) so standardizing was vital. Since there are over 30 features due to one hot encoding the skew values are located in the appendix. To deal with this, we scaled the features to a uniform distribution, giving them a mean of zero and variance of 1, to prevent negatives in our features, we also used the minmax scaler. This really helps with reducing bias in features having stronger weight. Finally, our data is cleaned and now we can split it into training and testing 75/25 split. The classifiers we used for this question were random forests, support vector machines, voting, bagging, AdaBoost, and decision trees.

2.3 Prediction of Legendary Pokemon

This question was dealt with as a supervised learning problem using classification of various techniques. Our predictor for classification was the 'isLegendary' column in our dataset and the features that were chosen to predict if a Pokemon was legendary or not was 'Total', 'Height_m', 'Weight_kg', 'Catch Rate' and 'has Gender'. The reason these were chosen was for a simple reason, through testing such as looking at correlation matrices, testing the models and looking for feature importance. These five features were the strongest predictors for Legendary by a large amount. Since there is no categorical data in this question, we do not have to worry about label encoding. The next and final thing we needed to worry about was to see if our data was skewed. Through using the skew() function in pandas, we were able to see if our features were either a negative, positive or no skew. 'Total' had little skew being a value of 0.06, 'hasGender' was a negative skew with a value of -2.56, 'Catch Rate' had a skew of 0.8 which is minimal and 'Height_m' and 'Weight_kg' had 5.51 and 4.01 respectively. To deal with this, we scaled the features to a uniform distribution, giving them a mean of zero and variance of 1, to prevent negatives in our features, we also used the minmax scaler. This really helps with reducing bias in features having stronger weight. Finally, our data is cleaned and now we can split it into training and testing 75/25 split. The classifiers we used for this question were logistic regression, random forests, support vector machines, voting, bagging, AdaBoost, and decision trees.

2.4 Prediction Body Style

Since this is a supervised learning problem, the data needs to be split into training and testing sets. Then implement this training and testing set into the methods. For normalizing the data I choose to use scaling. Scaling features to a uniform scale can help the algorithm treat all features equally (Figure 6.). The features being used for making the predictions are: 'Total', 'HP', 'Attack', 'Defense','Sp_Atk', 'Sp_Def', 'Speed', 'Height_m', 'Weight_kg','Catch_Rate','Type_1','Type_2',Egg_Group_1',Egg_group_2'. For the four features "Type_1" , "Type_2", "Egg_group_1", and "Egg_group_2" were one-hot encoding to create binary features for each category, which can help machine learning algorithms better understand the relationships between categories. These features were applied in my Deep Learning, K-Nearest, Select Vector Machine, Random Forest, Decision Tree, and Ensemble Hard/Soft Voting.

2.5 Clustering and standardization of the data and data cleaning.

We first started out by cleaning the data. Next in this case k-means clustering and dendrogram were used; both of these are an unsupervised learning problem. In addition to that the data was standardized to make it easier to read.

3. Results

3.1 Accurately predicting catch rate

3.1.1 Distribution of Catch Rate

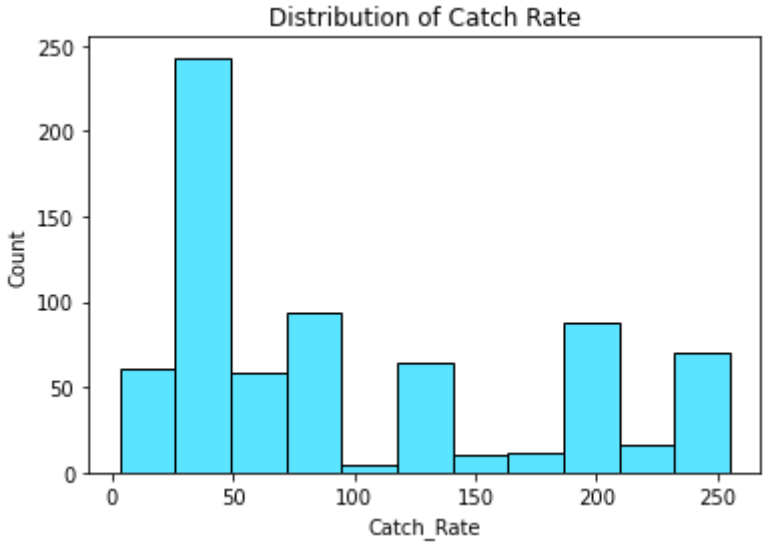

Figure 1. This figure shows the distribution of catch rate of pokemon.

3.1.1 Multiple Linear Regression

OLS Regression Results

Dep. Variable:	Catch_Rate	R-squared:	0.554
Model:	OLS	Adj. R-squared:	0.547
Method:	Least Squares	F-statistic:	78.61
Date:	Tue, 25 Apr 2023	Prob (F-statistic):	4.45e-104
Time:	23:36:23	Log-Likelihood:	-3432.6
No. Observations:	644	AIC:	6887.
Df Residuals:	633	BIC:	6936.
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	357.2773	10.972	32.561	0.000	335.731	378.824
HP	-0.7093	0.098	-7.263	0.000	-0.901	-0.518
Attack	-0.4229	0.095	-4.435	0.000	-0.610	-0.236
Defense	-0.5300	0.096	-5.540	0.000	-0.718	-0.342
Sp_Atk	-0.5429	0.093	-5.865	0.000	-0.725	-0.361
Sp_Def	-0.4694	0.104	-4.518	0.000	-0.673	-0.265
Speed	-0.5480	0.088	-6.195	0.000	-0.722	-0.374
Generation	1.9657	1.201	1.637	0.102	-0.392	4.323
Height_m	-3.5593	2.889	-1.232	0.218	-9.233	2.114
Weight_kg	0.0503	0.046	1.096	0.273	-0.040	0.141
Pr_Male	-70.2573	10.377	-6.770	0.000	-90.635	-49.879

Figure 2. This figure shows the summary output of a multiple linear regression model on the dataset.

3.1.2 Voting Regressor

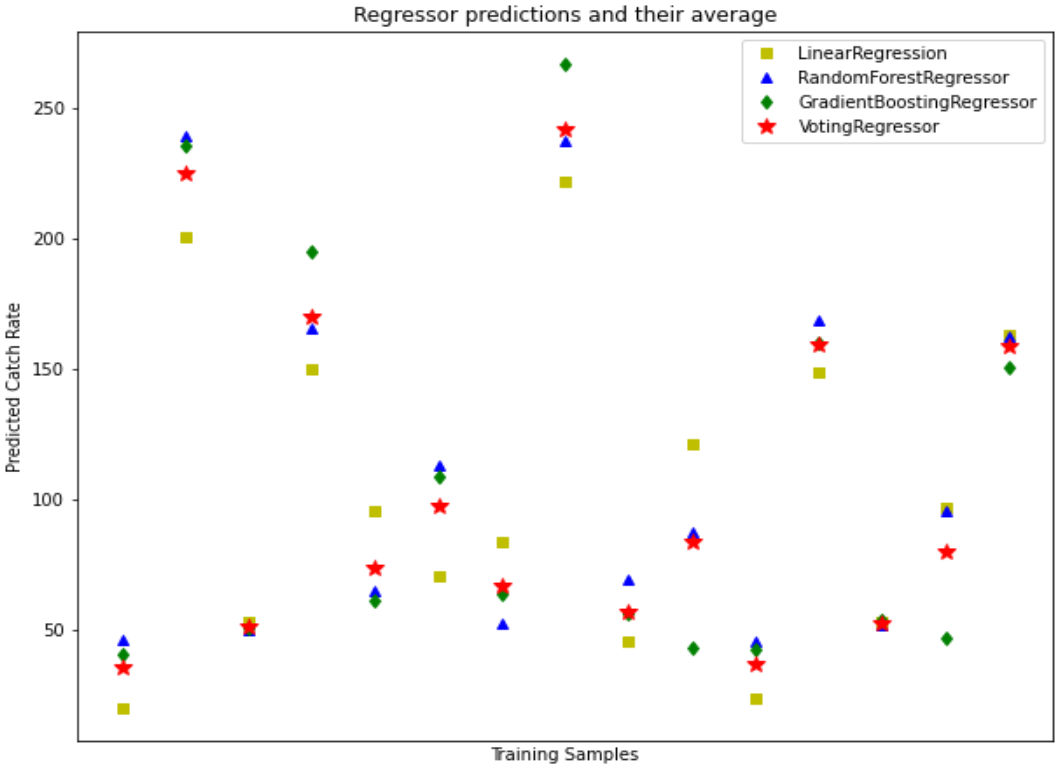

Figure 3. This figure shows a random sample of 14 pokemon and shows how each model predicts the catch rate of that pokemon.

3.1.3 Conclusion

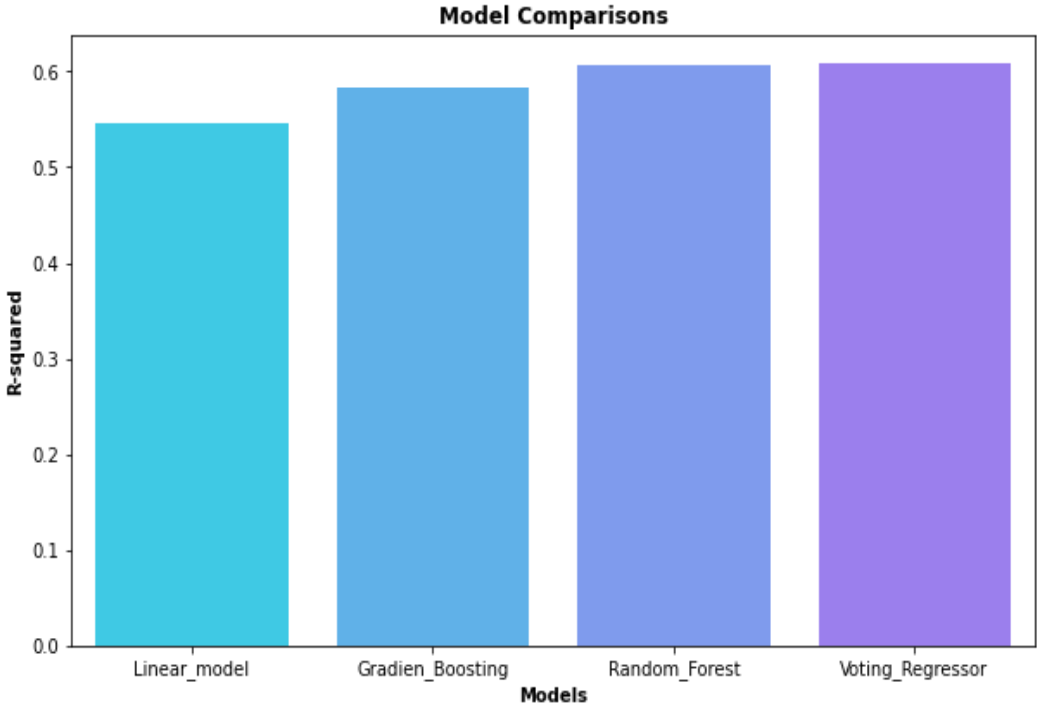

Figure 4. This figure shows all of the different regression models that were trained to the data set and their respective R-squared values.

3.2 Predicting Pokemon Type

3.2.1 Comparison of Classifier Accuracy

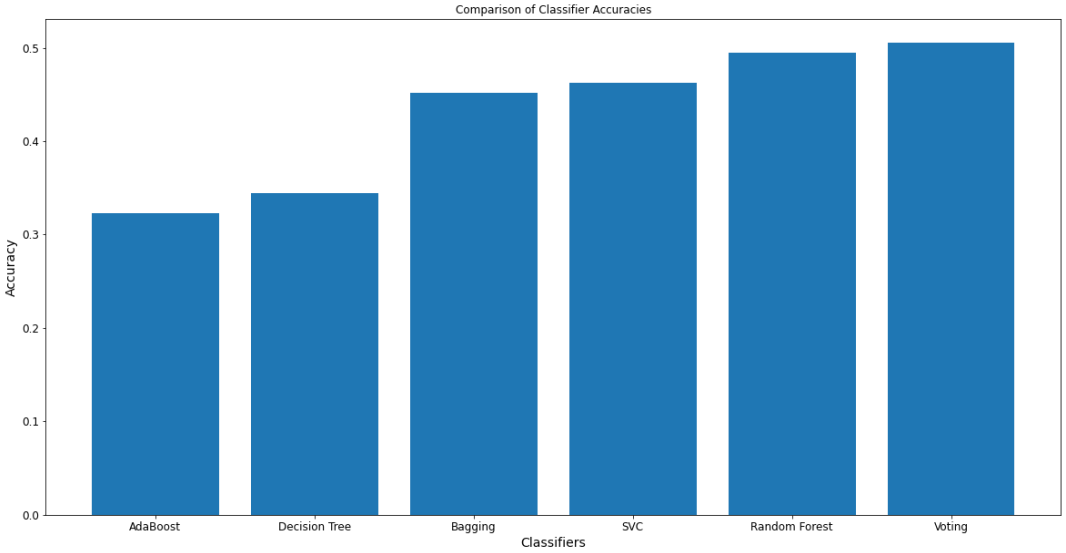

Figure 5. This figure shows the respective accuracy between all of the selected classifiers for predicting a type of Pokemon.

3.3 Predicting if Pokemon is Legendary

3.3.1 Feature Importance

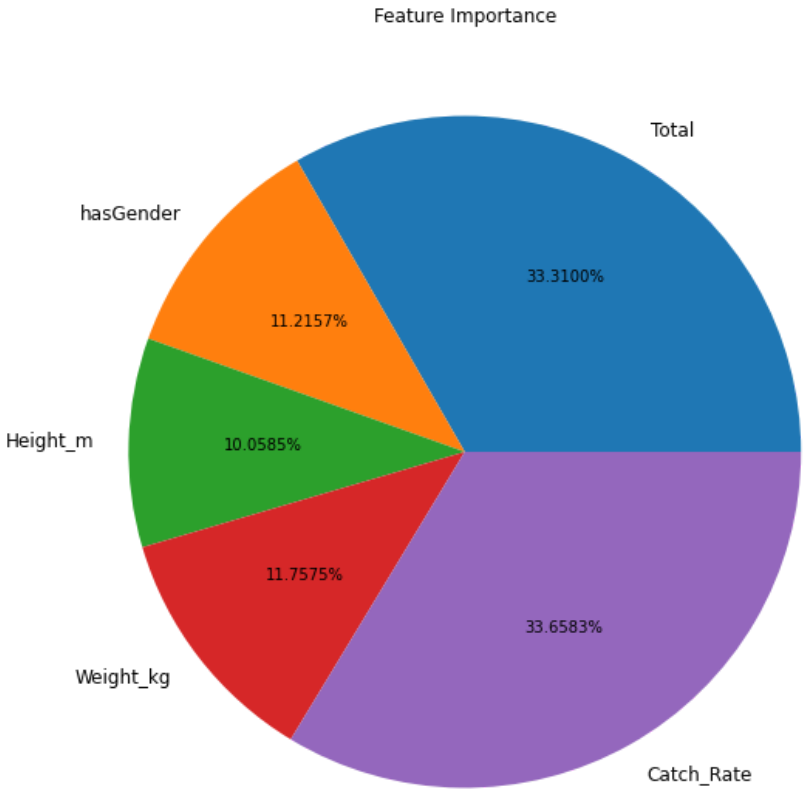

Figure 6. This figure displays the proportion of importance in random forest for each feature of the finalized predicting legendary dataframe.

3.4 Predicting Pokemons body style

3.4.1 Distribution of features before and after normalizing

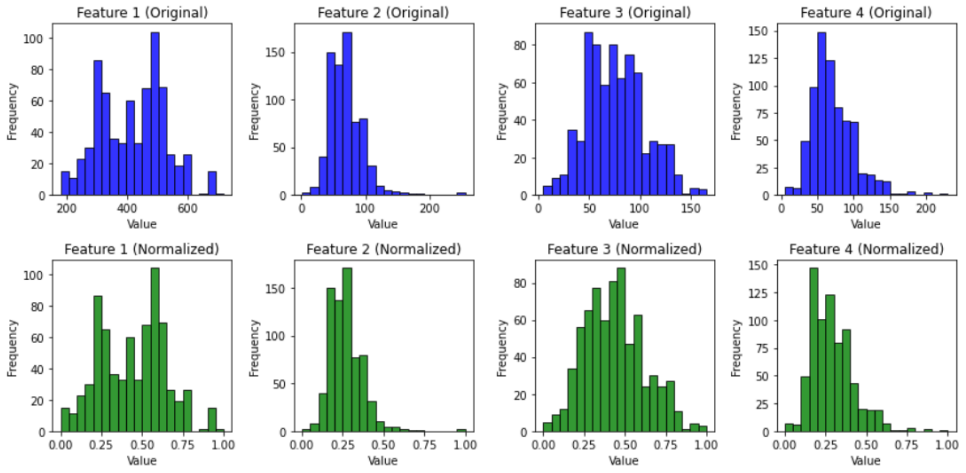

Figure 6. For problem "Prediction of body style" distribution for the first four out of fourteen features being normalized, which are 'Total', 'HP', 'Attack', and 'Defense'.

3.4.1 Comparison of Model Accuracy for predicting body style

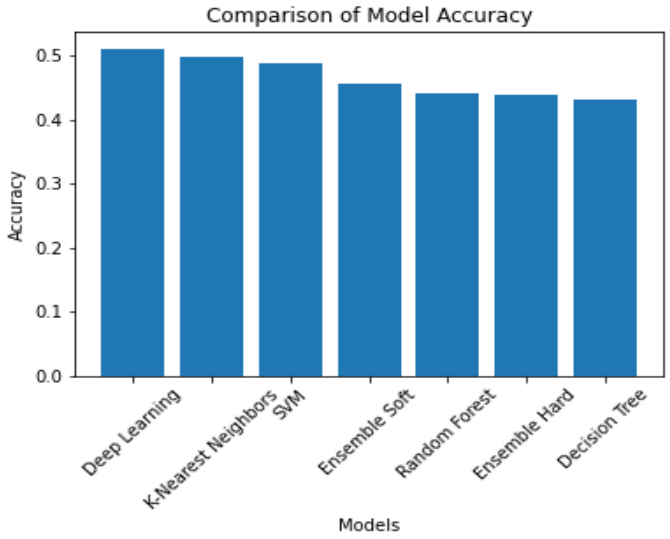

Figure 7. Showing the results of how the models perform from predicting body style. Deep learning is the best and Decision tree being the worst.

3.5 Clustering

3.5.1 Correlation of Numerical values

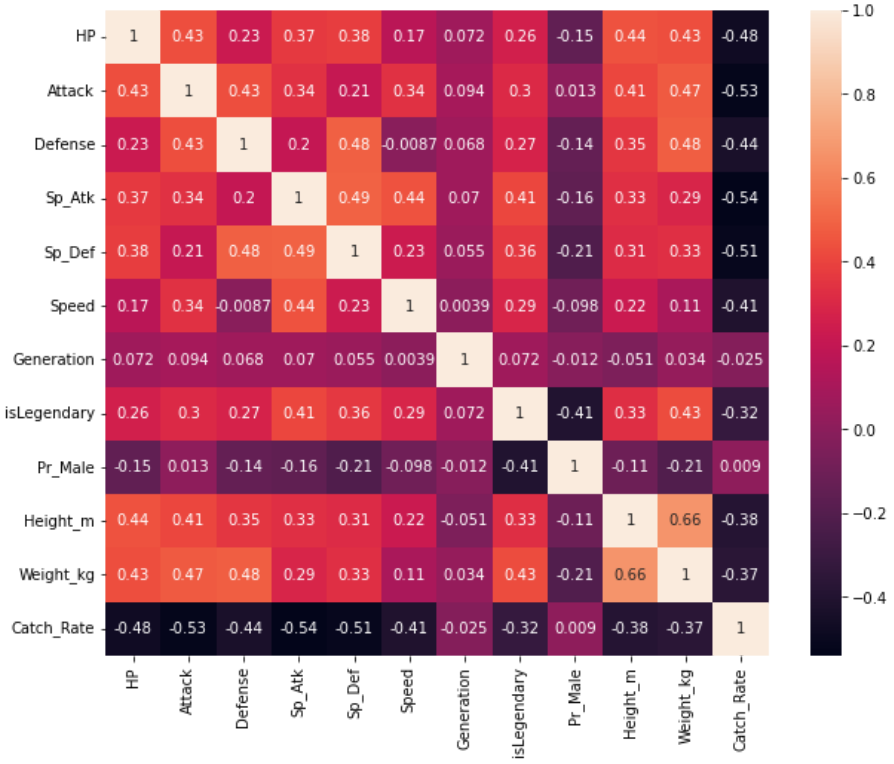

Figure 8. This figure shows the correlation between numerical values in the dataset.

3.5.2 Standardizing data

	Pr_Male	Generation	Total	HP	Attack	Defense	Sp_Atk	Sp_Def	Speed	Pr_Male	Height_m	Weight_kg	Catch_Rate
0	1.628168	-1.430397	-1.117514	-1.199832	-0.980330	-1.077927	-0.176955	-0.246667	-0.626658	1.628168	-0.550801	-0.716665	-0.749981
1	1.628168	-1.430397	-0.183023	-0.481623	-0.515742	-0.506978	0.442591	0.450931	-0.021363	1.628168	-0.188255	-0.623521	-0.749981
2	1.628168	-1.430397	1.105930	0.475990	0.199009	0.308665	1.268652	1.381060	0.785697	1.628168	1.132449	0.704928	-0.749981
5	1.628168	-1.430397	1.202602	0.380229	0.270484	0.104754	1.640380	0.683463	1.592757	1.628168	0.731059	0.559868	-0.749981
33	2.331868	-1.430397	0.891105	0.523870	0.913760	0.063972	0.649106	0.218398	0.987462	2.331868	0.342616	0.124686	-0.749981
45	-0.482931	-1.430397	-1.471976	-1.678639	-0.229841	-0.833235	-1.003017	-0.711732	-1.433718	-0.482931	-1.081673	-0.739569	1.445628
46	-0.482931	-1.430397	-0.183023	-0.481623	0.663597	0.186318	-0.383471	0.450931	-1.231953	-0.482931	-0.188255	-0.371573	-0.295717
78	-0.482931	-1.430397	-1.149738	0.954796	-0.408529	-0.425413	-1.209532	-1.409329	-1.837248	-0.482931	0.070707	-0.272321	1.445628
79	-0.482931	-1.430397	0.729986	1.194199	-0.051154	1.409782	1.268652	0.450931	-1.231953	-0.482931	0.601578	0.376633	-0.295717
82	-0.482931	-1.430397	-0.752311	-0.864668	-0.408529	-0.833235	-0.466077	-0.386186	-0.021363	-0.482931	-0.447217	-0.592982	-0.749981
86	-0.482931	-1.430397	0.568867	0.954796	-0.229841	0.186318	0.029560	1.148528	0.382167	-0.482931	0.731059	1.010319	-0.295717
110	-0.482931	-1.430397	-0.827500	0.475990	0.306222	0.798050	-1.622563	-1.874394	-1.433718	-0.482931	-0.188255	0.933971	0.385679
111	-0.482931	-1.430397	0.676279	1.673006	1.914411	1.817603	-1.003017	-1.176796	-0.828423	-0.482931	1.002969	1.010319	-0.522849
129	-0.482931	-1.430397	1.267050	1.194199	1.735723	0.145536	-0.383471	1.381060	0.826050	-0.482931	6.946139	2.766314	-0.749981
130	-0.482931	-1.430397	1.213343	2.870022	0.306222	0.186318	0.649106	1.148528	-0.021363	-0.482931	1.753957	2.537271	-0.749981
137	1.628168	-1.430397	-0.720087	-1.678639	-1.301967	1.001960	0.855622	-0.711732	-1.030188	1.628168	-0.939244	-0.707503	-0.749981
138	1.628168	-1.430397	0.783692	-0.002817	-0.587217	2.021513	1.888198	-0.014134	-0.223128	1.628168	-0.188255	-0.287591	-0.749981
139	1.628168	-1.430397	-0.720087	-1.918042	0.127534	0.594139	-0.589986	-1.176796	-0.223128	1.628168	-0.809763	-0.646425	-0.749981
140	1.628168	-1.430397	0.783692	-0.481623	1.378348	1.205871	-0.176955	-0.014134	0.785697	1.628168	0.213136	-0.203609	-0.749981
148	-0.482931	-1.430397	1.911526	1.002677	2.057361	0.798050	1.268652	1.381060	0.785697	-0.482931	1.391411	2.384576	-0.749981

Figure 9. This figure shows that we standardized the data to make it easier to read

3.5.2 Cleaning Values

	Number	Name	Type_1	Type_2	Total	HP	Attack	Defense	Sp_Atk	Sp_Def	...	Color	hasGender	Pr_Male	Egg_Group_1	Egg_Group_2	hasF
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	...	Green	True	0.875	Monster	Grass	
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	...	Green	True	0.875	Monster	Grass	
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	...	Green	True	0.875	Monster	Grass	
5	6	Charizard	Fire	Flying	534	78	84	78	109	85	...	Red	True	0.875	Monster	Dragon	
33	34	Nidoking	Poison	Ground	505	81	102	77	85	75	...	Purple	True	1.000	Monster	Field	
45	46	Paras	Bug	Grass	285	35	70	55	45	55	...	Red	True	0.500	Bug	Grass	
46	47	Parasect	Bug	Grass	405	60	95	80	60	80	...	Red	True	0.500	Bug	Grass	
78	79	Slowpoke	Water	Psychic	315	90	65	65	40	40	...	Pink	True	0.500	Monster	Water_1	
79	80	Slowbro	Water	Psychic	490	95	75	110	100	80	...	Pink	True	0.500	Monster	Water_1	
82	83	Farfetch'd	Normal	Flying	352	52	65	55	58	62	...	Brown	True	0.500	Flying	Field	
86	87	Dewgong	Water	Ice	475	90	70	80	70	95	...	White	True	0.500	Water_1	Field	
110	111	Rhyhorn	Ground	Rock	345	80	85	95	30	30	...	Grey	True	0.500	Monster	Field	
111	112	Rhydon	Ground	Rock	485	105	130	120	45	45	...	Grey	True	0.500	Monster	Field	
129	130	Gyarados	Water	Flying	540	95	125	79	60	100	...	Blue	True	0.500	Water_2	Dragon	
130	131	Lapras	Water	Ice	535	130	85	80	85	95	...	Blue	True	0.500	Monster	Water_1	
137	138	Omanyte	Rock	Water	355	35	40	100	90	55	...	Blue	True	0.875	Water_1	Water_3	
138	139	Omastar	Rock	Water	495	70	60	125	115	70	...	Blue	True	0.875	Water_1	Water_3	
139	140	Kabuto	Rock	Water	355	30	80	90	55	45	...	Brown	True	0.875	Water_1	Water_3	
140	141	Kabutops	Rock	Water	495	60	115	105	65	70	...	Brown	True	0.875	Water_1	Water_3	
148	149	Dragonite	Dragon	Flying	600	91	134	95	100	100	...	Brown	True	0.500	Water_1	Dragon	

Figure 10. This figure shows the null values were removed from the dataset cleaning it.

3.5.3 Denogram for Clustering

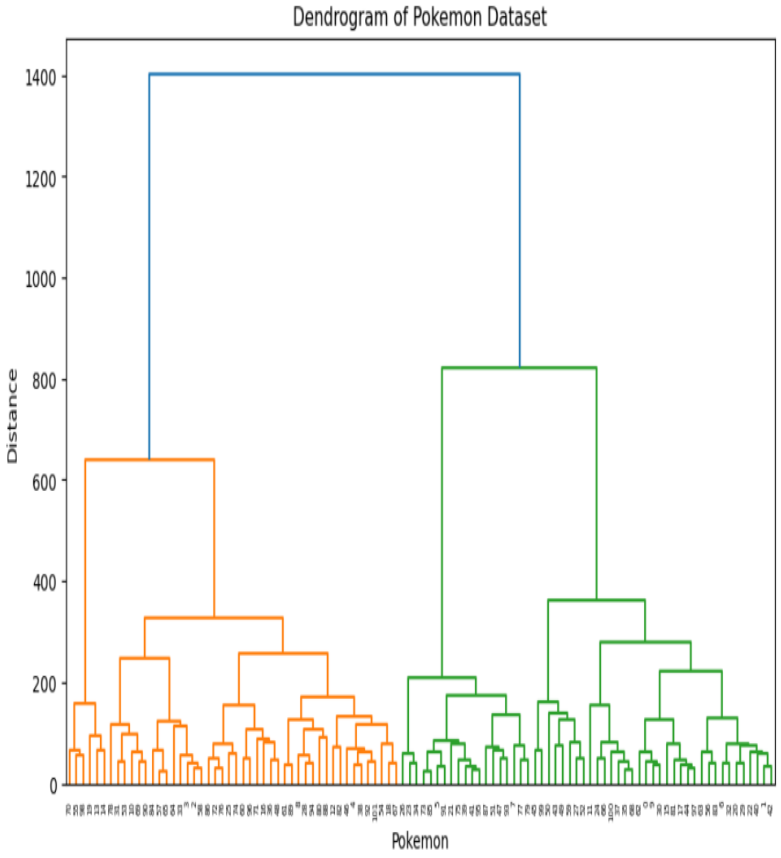

Figure 11. This denogram uses the Wards method and shows the distance of all the numericals within the dataset and from there it is broken up into subgroups.

3.5.3 K-Means clustering

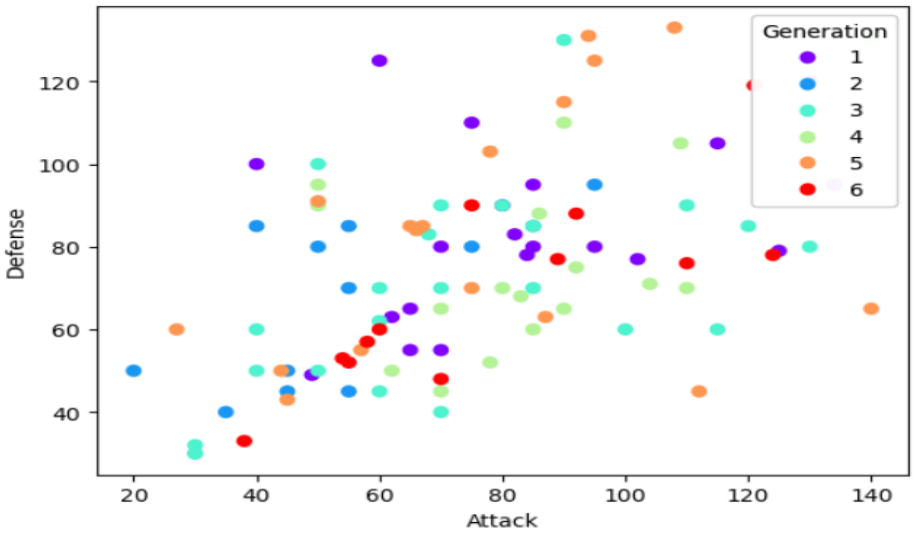

Figure 12. Shows K-Means clustering in correlation between the attack and defense and its generation

4. Discussion

4.1 Accurately Predicting catch rate of pokemon

4.1.1 Introduction

Since we are on a team looking to build the stats for new pokemon, it would be convenient to have a model that could accurately predict the catch rate of our pokemon. The methods that will be used to build the best algorithm for this is multiple linear regression, random forest regression and gradient boosting regression. Using these different models will help us to find which regression algorithm fits best to our data in terms of predicting, and also finding which method yields the most interpretable results. To start this process, it is important to check the distribution of our target variable. looking at figure 1, we can see that the data is slightly right skewed, so a Min-Max scaler from Sklearn will be used to normalize the data. Looking at figure 8, we can see that many of our numerical attributes are correlated with catch rate, so to start the modeling process, all of these predictors will be used.

4.1.2 Multiple Linear Regression

Analyzing Figure 2, we can see that the final linear regression model that the team came up with includes the predictors, HP, Attack, Defense, Special Attack, Special Defense, Speed, Generation, Height in meters, Weight in kilograms and the probability of it being male. The multiple R-squared came out to 0.554, so we can say that the model explains 54.4% of the variance in catch rate. To further interpret the model, as the probability of being male increases, the expected catch rate will drop substantially. As all of the battling stats increase, the expected catch rate decreases. As generation increases, so does the expected catch rate.

4.1.3 Random Forest and Gradient Boosting

After implementing a Random Forest Regressor with 1000 trees and a depth of 10 nodes per tree, we came up with an R-squared of .607. The Gradient Boosting Regression model yielded an R-squared of 0.584. Both of these models performed better than the previous Linear Regression model, but only by a small amount.

4.1.4 Voting Regressor

Analyzing Figure 3, we can see that these models seemed to predict pretty similarly to each other. The voting regressor was built using the past three methods explained and incorporates hard voting and an R-Squared of .608. This is the highest performing model, however it's R-squared is only 0.001 higher than the Random Forest model.

4.1.5 Conclusion

Analyzing Figure 4, we can see that the voting regressor was our best performing model. That being said, the interpretation of the linear regression model was by far the most superior and the R-squared is not that much lower in comparison to the voting regressor, so if the team was looking for interpretation, then use the linear model, but if the team is looking for the most accurate predictions of catch rate, then use the voting regression model.

4.2 *Predicting Pokemon Type*

4.2.1 Introduction

As researchers for Pokemon, it would be important to know if there is any correlation between any of our features for a given Pokemon and its type. To be able to accurately predict a Pokemon's type based on various features would mean that for future pokemon, we can design their qualities around the given type we want them to have. Classification will be used to deal with prediction and the specific models that are used are support vector machine, decision tree, random forests, AdaBoosting, bagging, and hard voting. We want to test all of these models to see which one produces the best accuracies in our prediction for type. The following features were used for every classifier: HP, Attack, Defense, Sp_Atk, Sp_Def, Speed, Generation, isLegendary, hasGender, height, weight, catch rate, color, body style

4.2.2 Support Vector Machine

Using the given features, and using this classifier's default parameters, we discovered that Support Vector Machine had an accuracy of 46.24% for predicting pokemon type.

4.2.3 Random Forests

Random Forests with the parameter of 100 estimates ended up with a 49.64% accuracy, being one of the closest accuracies we will get between all classifiers. When deciding on which classification model to use for predicting types, we would most likely choose this or the next classifier.

4.2.4 Hard Voting

Using the previous two classifiers as the foundation for hard voting as well as decision tree, the accuracy ended up being the highest at 50.54%. This would mean that we would want to pick this classifier most likely as our final model. We can safely say that the model is able to predict a Pokemon's type half of the time.

4.2.5 AdaBoosting

AdaBoosting had a very low accuracy of 32.26% using 200 samples. Even with more or less samples, accuracy was still surprisingly low. We would definitely avoid this when considering what model is the best for classification.

4.2.6 Bagging

Bagging with bootstrap had a decent accuracy of 45.16%, while it is not the highest accuracy, it is not the lowest.

4.2.7 Decision Tree

Decision Tree was very low as well, being close in accuracy to AdaBoosting at 34.41%. We would not recommend using this model as a classifier.

4.2.8 Conclusion

Looking at Figure 5, we can see that our best model was hard voting, with random forest being a very close second choice. Most other classifiers such as SVM or Bagging ended up being around the same range of mid 40's. AdaBoost and Decision Tree ended up being the worst performance, where they were in the 30's. We would probably not use AdaBoost or Decision Tree considering how many other options produced better results. Only one of these models had accuracy that broke above 50%, this is very low considering we want accuracy closer to 100%. The models, while having low percentages, cannot really be considered weak learners as they are still significantly greater than a random guess of a Pokemon's type (1/18 chance). Technically because a random guess would be .05% likely, our best model increases the likelihood of guessing a pokemon correctly by 100 times. Ultimately, we can most likely conclude that there are just too many types of Pokemon for our models to accurately predict a Pokemon's type and also that many of these features that a Pokemon has, don't really have strong enough correlations to type to show any trends.

4.3 *Predicting if Pokemon is Legendary*

4.3.1 Introduction

Legendary Pokemon are very special to the game series, being a selling point for every edition of the game. They often are very rare to find in the games as well as have very powerful movesets which make players want to seek them out to capture. It would be very important to be able to predict if a pokemon is legendary or not based on certain stats to know how powerful it should be and how hard it is to catch. The features we ended up using for prediction are total stats, catch rate, gender, height and weight. These were determined through numerous feature importance tests and these five ended up being the most significant for isLegendary as a predictor. Looking at figure 5, we can see that catch rate and total stats make up a great deal of importance while the remaining features still make up a great deal.

4.3.2 Logistic Regression

Using the finalized features for this prediction, and the classifier's solver being newton-cg and iterations being 1,000, our accuracy for this model was 98.90%. This model was extremely accurate and there would be no reason not to use this as a model. Having high accuracy will not be uncommon through these comparisons.

4.3.3 Support Vector Machine

Using the same features, and using the default parameters, we discovered that Support Vector Machine had an accuracy of 97.79%% for predicting pokemon type. This is really good and not too far off from logistic regression.

4.3.4 Random Forests

Random Forests with the parameters of 4.2.4 estimates ended up with a 100% accuracy, being the same as Logistic Regression and the following classifiers.

4.3.5 Hard Voting

Using the previous three classifiers as the foundation for hard voting, the accuracy ended up being exactly the same as Logistic Regression. This could mean that the individual classifier votes chose logistic regression as the ideal model which is interesting as random forest has a higher accuracy.

4.3.6 AdaBoosting

AdaBoosting, using the same parameters as from 4.2.6, produced 100% accuracy in predicting a legendary pokemon.

4.3.7 Bagging

Bagging with the same parameters as from 4.2.7 had a 100% accuracy just like the past few questions has had, this is a very good trend.

4.3.8 Decision Tree

Decision Tree was also 100%, surprisingly ended up not being one of the weaker models.

4.3.9 Conclusion

Based on all of these model accuracies, it can be concluded that our specific models were highly accurate in predicting if a Pokemon is legendary or not. All classifiers except for Support Vector Machine, logistic regression and hard voting were 100% accurate. Even then they were all very accurate, only being off by a little. It can be inferred that the features we chose were very significant to our prediction and also since our predictor has only two choices, it could have a part in having such high accuracy across all classifiers. There could be a case of overfitting occurring, but because of pre-processing such as standardizing the data and EDA, as well as comparing test and training accuracies (they were extremely similar), we know this is most definitely not an issue that needs to be brought up.

4.4 *Prediction of Body Style*

4.4.1 Introduction

The prediction of body style was done through many different models and methods. The data was normalized by scaling the features and splitting the data accordingly. The founding of the data is shown in Figure 7. where the highest model, Deep learning is able to predict a pokemons body style 51% of the time. The worst performing model is the decision tree with its accuracy being 43%. Although these models might have not have a accuracy rate above 51% it is good to note that the body styles in total are 14, so if randomly chosen there is a 1/14 = 0.07 = 7% chance of randomly choosing the correct body style for a pokemon. As for the difference between models are:

4.4.2 Deep Learning

Deep learning models, especially neural networks, are capable of learning complex, non-linear relationships between features and the target variable. Thus the deep learning models were better suited to capture those patterns.

4.4.3 K-Nearest and SVM

For k-NN and SVM these models might have been sensitive to the scale of the features and may not perform well if the features have different scales or if the classes are not well-separated.

4.4.4 Random Forest and Decision Tree

Random forests and decision trees struggled because the features and the target variable are not well represented by the tree structure.

4.4.5 Hard and Soft Voting

The base models used in the ensemble have poor performance, which resulted in the ensemble's performance being worse than deep learning. Ensemble methods rely on combining strong base models to improve overall performance.

4.5 Clustering

4.5.1 Introduction

Through the use of unsupervised learning algorithms my goal was to see if there were any significant patterns. I did this in two ways firstly by using a dendrogram of and secondly using K-Means clustering.

4.5.2 Seeing if attack and defense power are correlated to generation.

Figure 12. Shows K-Means clustering in correlation between the attack and defense and its generation. As for this figure it is fair to say that when this K Means clustering function was runned it divulged that there was no discernable pattern between attack power defense and the generation of the pokemon. Based on that I can deduce that as stated earlier there is no correlation to speak of.

4.5.3 Dendrogram of Pokemon Dataset.

Figure 11. This dendrogram uses the Wards method and shows the distance of all the numericals within the dataset and from there it is broken up into subgroups. Doing further analysis it was found that there is no discernible pattern between the clustering. all the variables were numericals. The only thing that is really relevant is the fact that the distance between the first and second big clusters are that of a distance of 200.

5. Conclusions

When it comes to predicting catch rate, we notice that as the probability of being male increases, we expect the catch rate to drop substantially. As stats increase, catch rate decreases. As generation increases, so does the catch rate. The best model produced was a voting regression model using multiple linear regression, Random Forests Regression, and Gradient Boosting regressor, and it had an R-squared of .608. When we looked at predicting a Pokemon’s type, we noticed that we did not find that strong of a

correlation between many of the features and a Pokemon's type. The best performing model was a random forest classifier at 49.64%. Most models were either around the 40s or 30s for percentage which is pretty low. This is still technically a big improvement over randomly guessing (1/18 chance) a Pokemon's type meaning that our models aren't necessarily weak learners. Similar to predicting body style however the best model was deep learning, since it had the best ability to find complex correlation between features unlike the other models. Although most of the models for body style were around 40% to 50% this is still better than having a 1/14 chance of randomly choosing the correct body style. Classification for finding if a Pokemon is legendary had very high accuracies across all classifiers. Almost every model that was chosen was either at or close to 100% accurate. We can safely say that for our five features, height, weight, catch rate, total stats, and gender are highly related to being legendary. As values such as catch rate go down and total stats go up, the more likely a pokemon is to be legendary. When it comes to the attack and defense power while k means clustering there was no sort of pattern when analyzed. In addition to that during the dendrogram analysis there was no sort of pattern either.

Author Contributions: "Conceptualization, M.B.; methodology, M.B., S.R., L.L., C.H.; validation, M.B., S.R., L.L., C.H.;formal analysis, M.B., S.R., L.L., C.H.; investigation, M.B., S.R., L.L., C.H.; resources, M.B., S.R., L.L., C.H.; data curation,M.B., S.R., L.L., C.H..; writing—original draft preparation, M.B., S.R., L.L., C.H.; writing—review and editing, M.B., S.R., L.L., C.H.; visualization,M.B., S.R., L.L., C.H.;All authors have read and agreed to the published version of the manuscript."

Data Availability Statement: The data is available online from kaggle. To find the repository click here.

References

1. All You Need to Know about Gradient Boosting Algorithms. Available online: https://towardsdatascience.com/
2. Scikit-learn documentation: https://scikit-learn.org/stable/index.html
3. Matplot lib documentation: https://matplotlib.org/stable/