Machine Learning

Aldo Barletta,
Gianni Eduard Balbin Canchanya,
Fabio Galante

June 2023

Contents

1 Introduction

In the world of Pokémon, legendary Pokémon hold a special place. These rare and powerful creatures possess unique abilities, captivating designs, and extraordinary lore, making them highly popular. However, the process of determining whether a Pokémon is legendary or not can be challenging, especially for researchers and trainers who aim to understand the underlying factors contributing to their legendary status.

This report presents a comprehensive analysis of Pokémon statistics, aiming to predict whether a Pokémon is legendary solely based on its attributes and characteristics. By examining a diverse range of Pokémons and their corresponding stat profiles, we aim to uncover patterns and trends that differentiate legendary Pokémon from their non-legendary counterparts. This research can potentially shed light on the defining factors that contribute to the extraordinary powers and status of legendary Pokémon.

In order to conduct this analysis, we are going to use these features:

- **Name**: The name of the Pokemon;
- **Type-1**: The primary type of the Pokemon;
- **Type-2**: The secondary type of the Pokemon (in case the Pokemon has it);
- **Total**: Sum of the basis stats (Health Point, Attack, Defense, Special Attack, Special Defense and Speed);
- **HP**: The base Health Points of the Pokemon;
- **Attack**: The base Attack of the Pokemon;
- **Defense**: The base Defense of the Pokemon;
- **Sp-Atk**: The base Special Attack of the Pokemon;
- **Sp-Def**: The base Specia Defense of the Pokemon;
- **Speed**: The base Speed of the Pokemon;
- **Generation**: Number of the generation when the Pokemon was introduced;
- **isLegendary**: Boolean that indicates whether the Pokemon is legendary or not;
- **Color**: Color of the Pokemon according to the Pokedex;
- **hasGender**: Boolean that indicates if the Pokemon

can be classified as male or female;
- **Pr-Male**: In case the Pokemon has Gender, the probability of its being male. The probability of being female is, of course, 1 minus this value;
- **Egg-Group-1**: Egg Group of the Pokemon;
- **Egg-Group-2**: Second Egg Group of the Pokemon, in case it has two;
- **hasMegaEvolution**: Boolean that indicates whether the Pokemon is able to Mega-Evolve or not;
- **Height-m**: Height of the Pokemon in meters;
- **Weight-kg**: Weight of the Pokemon in kilograms;
- **Catch-Rate**: Catch Rate of the Pokemon;
- **Body-Style**: Body Style of the Pokemon according to the Pokedex.

2 Data Exploration

The dataset downloaded from Kaggle includes data up to the seventh generation of Pokémon, with a total of 721 observations. Each generation features some Pokémon considered legendary within the Japanese video game.

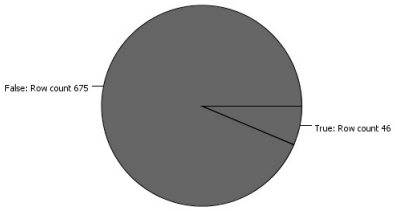

Figure 1: *Pie Chart of the Pokemons*

A total of 46 legendary Pokémon were found, representing approximately 6 percent of the entire dataset. This relatively small proportion highlights the rarity and exclusivity of legendary Pokémon. These extraordinary creatures are highly famous and renowned for their exceptional powers and significance within the Pokémon universe.

Given our research objective, we will designate the "isLegendary" attribute as the dependent variable in our analysis. This binary variable, with its two possible outcomes, provides a clear distinction between legendary and non-legendary Pokémon. By considering this attribute as our focus, we aim to uncover valuable insights into the factors that differentiate legendary Pokémon from their non-legendary counterparts.

3 Preprocessing

3.1 Feature Selection

In this case, there's been no proper feature selection, given the fact that our research has the purpose to find a Pokemon considered peculiar and rare which has certain characteristics, we consider all the attributes inside our dataset as relevant and fundamental.

3.2 Feature transformation

We know that legendary Pokémon do not have a gender, but there are also some Pokémon that possess this characteristic without being considered legendary. In our dataset, we have an attribute called "*Pr-Male*" that indicates whether a Pokémon is male or not. Importantly, this attribute contains some probabilities that are zero and some that are denoted with "?".

To address this, we have created a variable called "*Pr-Female*", which represents the probability of a Pokémon being female. This variable is obtained by taking the complement of 1 from the "*Pr-Male*" attribute. By doing so, we account for the possibility of a Pokémon being female, considering that the "*Pr-Male*" attribute focuses on male probabilities.

$$\textbf{Pr-Female}_i = 1 - \textbf{Pr-Male}_i$$

where i = correspondent value. We are adding this new attribute because there are missing values in this column, indicating Pokémon without a gender.

Later on, we will replace the missing values with a consistent value in both columns. This will indicate that the Pokémon has a probability equal to zero of having the gender specified in the column we are analyzing and, as a result, is genderless.

3.3 Handling of missing values

In the descriptive analysis, we have identified that four attributes contain missing observations: **Type-2**, **Egg-Group-2**, **Pr-Male**, and **Pr-Female**. **Type-2** and **Egg-Group-2** are string type attributes, while **Pr-Male** and **Pr-Female** are of double type.

To address these missing values, we have opted for a replacing method.

For the string attributes, we have decided to replace the "?" values with "*None*" values, because we already know that there's a possibility a Pokemon does not have a second type.

For the *double* attributes, we have replaced the missing values with "0.0", denoting a probability of zero for gender-related attributes.

This approach ensures that if a user is searching for genderless Pokémon, they can filter the dataset based on these two gender probabilities to find the desired information. By replacing the missing values with appropriate placeholders, we enhance the usability and integrity of the dataset for further analysis.

4 Methods

We have used different kinds of Machine Learning models in order to predict whether a Pokemon is legendary or not, and they are:

- **J48**, it allows classification via either decision trees or rules generated from them.
- **Naive Bayes**, it is based on Bayes theorem and it is mainly used in text classification.
- **Support Vector Machine** (SVM), it is used for Classification as well as Regression problems.
- **Logistic Regression**, it is the most used method for binary classification problems.
- **K-Nearest Neighbour** (KNN), it is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems.
- **Random Forest**, it is a supervised machine learning algorithm that is constructed from decision tree algorithms.

All of the models above are based on *Weka 3.7*, apart from the KNN model.

The structure of our workflow, after cleaning the dataset, is primarily organized around creating techniques to evaluate the performance of the aforementioned models. Once we have evaluated the most efficient model for each technique, we will compare the best methods obtained to determine the ultimate best model.

It's important to note that for each model we choose 5 performance methods and they are:

- **Holdout**
- **Iterated Holdout**
- **K-Fold Cross Validation (Random)**
- **K-Fold Cross Validation (Stratified)**
- **Leave-One-Out Cross-Validation (LOOCV)**

4.1 Holdout

The *Holdout* approach for machine learning is commonly used to split the dataset into a training set and a test set. In this case, we will randomly sample the dataset and allocate 2/3 of the data for the training set and the remaining 1/3 for the test set.

By setting the seed to *5678*, we ensure that the random sampling process is reproducible, meaning we can obtain the same train-test split every time we run the code with this seed value. This can be useful when comparing different models, as it allows for fair comparisons and reproducibility of results.

This approach helps us estimate how well the model will generalize to new, unseen instances.

By splitting the dataset randomly, we aim to create two sets of data that are representative of the overall dataset. The training set provides the necessary examples for the model to learn patterns and make predictions, while the test set serves as an unbiased evaluation of the model's performance.

4.2 Iterated Holdout

The *Iterated Holdout* approach extends the Holdout method by repeating it multiple times to obtain a more robust evaluation of a machine learning model's performance. In this case, we will iterate the Holdout method a total of $R = 5$ times.

For each iteration $r = 1,...,5$, we will perform the Holdout method with a random sample. Even in this case, the random sample D_T will consist of 67 percent of the records from the original dataset. This means that in each iteration, we will have a different training set and test set, both derived from random sampling.

By repeating the Holdout method multiple times, we can obtain a more reliable estimate of the model's performance. This helps to mitigate the potential bias or variability that can arise from a single train-test split.

Once we have performed this method for all R iterations, we can aggregate the evaluation results to obtain a more comprehensive assessment of the model's performance. This can include metrics such as accuracy, precision, recall, or any other relevant measures.

4.3 Cross Validation

In the Cross-Validation approach, we partitioning the dataset D into multiple folds. This helps in obtaining a more robust evaluation and understanding of the model's generalization capabilities.

To ensure fairness and avoid biases in the evaluation, it is important that each record in the dataset D is included in the training sets the same number of times and exactly once in the test set. This means that each record should have an equal chance of being in the test set and should not be included in multiple test sets simultaneously.

To achieve this, the dataset D is partitioned into 10 disjoint subsets. Each subset contains almost an equal number of records, resulting in a nearly constant number of records in each partition.

In this case, we used two different sampling methods:

- **Random Sampling**: the records in the dataset D are randomly assigned to each of the 10 partitions. This random assignment ensures that each record has an equal chance of being included in any subset.
- **Stratified Sampling**: involves dividing the dataset into groups based on specific characteristics and then randomly sampling within each group to maintain proportional representation.

4.4 LOOCV

Leave-one-out cross-validation is a method used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model. In order to do so LOOCV uses the following approach to evaluate a model: first, this method splits the given dataset into a training set and a testing set, using all but one observation as part of the training set, therefore we only leave one observation "out" from the training set (this is where the method gets the name "leave one out" cross validation).

After this, LOOCV builds the model using only data from the training set in order to use it as a way to predict the response value of the observation, which is left out of the model. This process is going to be repeated n times, where n is the total number of observations in the dataset, leaving out a different observation from the training set each time.

5 Evaluation

Now we analyze how the methods previously seen perform on each model.

5.1 Holdout

5.1.1 Recall

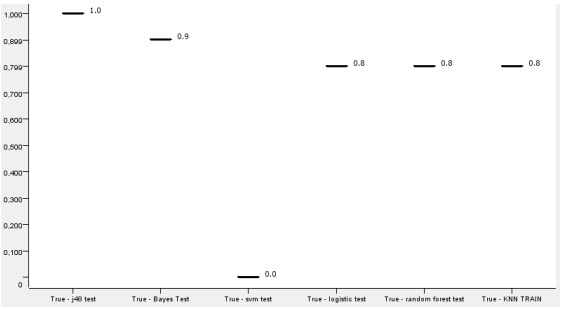

Figure 2: *Recall of the test set with the Holdout method*

In the recall measure, which tells us the portion of records correctly predicted by the model, we see that in the train model the obtained values were all optimal even though the *J48* model has got a value equal to 0.97.

In the test set, though, we have a different story: in fact, in the Figure 2, the model which predicts the true records better than the other is *J48*, while the other models drop in the effectiveness in prediction. It's worth to mention that the *Support Vector Machine* model has a recall equal to 1 in the train set while equal to 0 in the test set, indicating that it has correctly predicted zero true records.

5.1.2 Accuracy

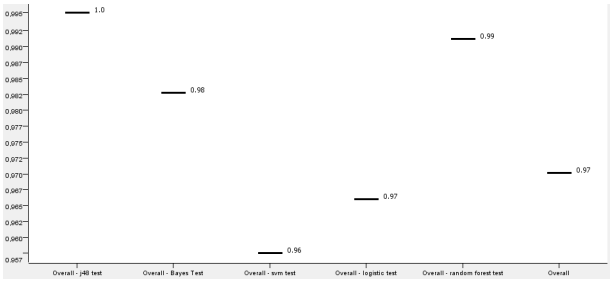

Figure 3: *Accuracy of the test set with the Holdout method*

The accuracy measures the ability of the model to correctly classify new records. We can see that it performs perfectly with the *Support Vector Machine, Logistic Regression* and *Random Forest* model. The lowest value, even though it is still pretty high, is equal to 0.98, which corresponds to the *Naive Bayes* model.

If we look at the Figure 3, which represents the accuracy of the test set, we can observe a different performance: here it is possible to notice that the best model for the test set there's been a change, now we have the *J48* model which performs better. The best models of the train set are still able to classify correctly but not perfectly, in fact we have a decreasing in performance, in particular the *Support Vector Machine* and the *Logistic Regression*, whose values are respectfully 0.06 and 0.97.

5.1.3 ROC

The ROC Curve allows us to understand which model performs better regardless of the class distribution, so we can see which model is more efficient than the other ones.

In the train set we refer that the models which perform perfectly are the *Support Vector Machine, Logistic Regression* and the *Random Forest* models, but the other classifiers perform good as well.

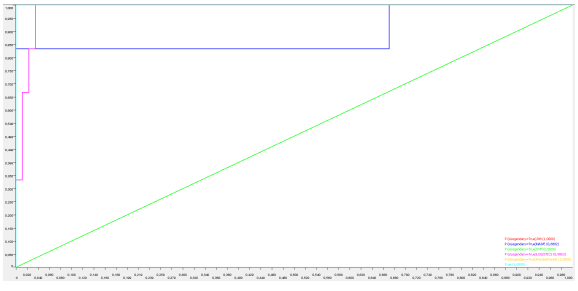

Figure 4: *ROC of the test set with the Holdout method*

In the test set, on the other hand, we have a drastic decline of performance by the *Support Vector Machine* model, which drops in half, as you can see in the Figure 4.

The models that stand out from the other ones are the *J48* and the *Random Forest*, therefore according to the ROC Curve, it would be logical assuming that the model which performs the best in both the train and the test sets is the *Random Forest* classifier.

In this case, though, we are going to pick as the best model the *J48* nonetheless, because in the measures previously seen we have noticed that this particular classifier is the best among the other ones.

5.1.4 Cost

The Cost values help us understand how much a given company has to pay in order to let a particular model to run up. Once again, we refer that in the train set the models which cost less to the company in order to run and perform various tasks are the *Support Vector Machine, Logistic Regression* and the *Random Forest* with a value equal to -483 overall. The *J48* and the *K-Nearest Neighbour* have little costs as well, while the *Naive Bayes* costs the most.

It is worth to mention that in the test set we have an increasing in the overall cost of the models, this can be due to several reasons like potential overfitting of the model in the test set. In this set, the situation has completely changed: we have as the model which costs the least is the *J48* that went from -451 to -230, while the other models' costs have increased.

In summary, the best model using the *Holdout* method is the *J48* model for the reasons above.

5.2 Iterated Holdout

5.2.1 Recall

We remember that for the *Iterated Holdout* method we iterate 5 times.

Row ID	True bayes	True j48	True svm	true logistic	true random	True knn
Iteration #0	1	0.857	0	0.929	0.933	0.909
Iteration #1	1	0.933	0	0.786	0.842	0.923
Iteration #2	1	0.538	0	0.75	0.812	0.923
Iteration #3	1	0.923	0	0.778	1	0.944
Iteration #4	1	0.9	0	0.824	0.947	0.917

Figure 5: *Recall of the test set with the Iterated Holdout method*

In the train set, the *Support Vector Machine, Logistic Regression*, and *Random Forest* models demonstrate perfect recall measurement. The *Naive Bayes* model also performs well, except for a decline in performance in the fourth iteration, resulting in a recall score of 0.97.

However, there has been a change in performance when evaluating the models on the test set. From Figure 5, we observe that the previously mentioned models have experienced a decline in performance. On the other hand, the *Naive Bayes* model remains the best-performing model, particularly achieving a perfect score in the third iteration, where there was a slight decline in the train set.

Anyway, the models show in both the sets optimal results.

5.2.2 Accuracy

Row ID	test j48	test bayes	test svm	test logistic	test random	test knn
Iteration #0	0.987	0.983	0.945	0.966	0.987	0.975
Iteration #1	0.975	0.983	0.937	0.966	0.983	0.983
Iteration #2	0.975	0.966	0.962	0.966	0.979	0.979
Iteration #3	0.983	0.979	0.941	0.958	0.992	0.971
Iteration #4	0.987	0.966	0.958	0.966	0.992	0.962

Figure 6: *Accuracy of the test set with the Iterated Holdout*

In the train test, like in the previous measurement, *Support Vector Machine, Logistic Regression* and *Random Forest* are still perfect models, after these there are the models *J48, K-Nearest Neighbour* and *Naive Bayes* in this order.

Regarding the test set, there has been a decreasing in the performance for every model but we are going to choose as the best model the *J48* and the *Random Forest* model.

5.2.3 ROC

The ROC curve in the train set presents values of performance that are quite close to 1. The models mentioned previously, which have demonstrated perfect performance in the measures discussed for the train set, continue to exhibit exceptional results.

However, when considering the ROC curve of the test set, all models achieve a perfect outcome. It is worth noting that the *SVM* model obtained a significantly lower result. Specifically, the ratio of *%TP* to *%FP* is at 50 percent.

This discrepancy in performance between the *SVM* model and the other models indicates a difference in their ability to accurately classify positive and negative instances.

5.2.4 Cost

Row ID	Cost_j48	Cost_Bayes	Cost_svm	Cost_logistic	Cost_random_forest	Cost_knn
Iteration #0	-214	-206	-134	-174	-214	-190
Iteration #1	-190	-206	-118	-174	-206	-206
Iteration #2	-190	-174	-166	-174	-198	-198
Iteration #3	-206	-198	-126	-158	-222	-182
Iteration #4	-214	-174	-158	-174	-222	-166

Figure 7: *Cost of the test set with the Iterated Holdout method*

On average, in the train set, the cost for each model and iteration is approximately lower by at least -450 units. One particular detail can be observed in the *Naive Bayes* model, where the cost is above -400 for some iterations. However, once again, the *SVM, Logistic Regression*, and *Random Forest* models exhibit significantly lower costs compared to others, with a result of -483.

In the test set, the iterated method behaves similarly to the holdout method analyzed previously, as there is an increase in cost for each model. In fact, the values are around -200 or above. Nevertheless, from Figure 7, we can observe that the *Random Forest* model consistently exhibits smaller costs compared to others.

For the *Iterated Holdout* method, considering the results obtained, we have decided to designate the *Random Forest* method as the reference model. This choice is primarily due to its inferior cost performance compared to other models.

5.3 Cross Validation

For the Cross Validation we noticed that the values between the *K-Fold*, *LOOCV* and *Stratified* are very similar with each other. We took the decision to consider only the Stratified Cross Validation because this type of method has got the best valuesamong the other types of Cross Validation methods, even if only slightly.

5.3.1 Recall

The models, namely *J48, Naive Bayes*, and *K-Nearest Neighbor*, stand out with exceptionally high recall values of 0.93, 0.98, and 0.93, respectively. These figures highlight their remarkable capability to effectively capture positive instances within the dataset.

The only exception is the *Support Vector Machine* model, where we have a recall equal to 0, therefore this type of classifier correctly predicts zero positive records.

5.3.2 Accuracy

We notice that the best model for the accuracy is the *J48* with a value equal to approximately 0.99, even though the difference between the other models is quite narrow. In this case, once again, the *Support Vector Machine* is the model with the lowest performance, even if it is still equal to 0.93.

5.3.3 Cost

Among the models considered, *Random Forest, K-Nearest Neighbor*, and *J48* demonstrate lower costs, with *J48* having the lowest value of -649.

The *Support Vector Machine* model stands out with the highest cost of -345. These cost variations suggest that the former models offer more cost-effective solutions compared to the latter, making them potentially preferable options in terms of resource allocation.

Based on its high recall value, lower cost, and superior performance compared to other models, we have determined that the *J48* model is the most suitable choice and, therefore, our selection for the best model.

6 Conclusion

We, then, compared the chosen models for the *Holdout*, *Iterated Holdout*, and Cross *Validation Stratified* methods, which, as a reminder, included the *Random Forest* for the second method and the *J48* for the other two methods.

From the results obtained for Recall, Accuracy, and Cost, it seems that the absolute best model for this research is the *J48* model from the *Cross Validation Stratified* method. Although the Holdout method shows the most efficient recall, the stratified approach still performs well with a recall rate of 0.93.

In conclusion, we ultimately consider the *Cross Validation Stratified* method with the *J48* model due to the excellent results obtained in accuracy and cost, namely 0.99 and -649, respectively.

Additionally, the *J48* model's interpretability allows users to gain insights into the decision-making process, making it a suitable choice for understanding the factors that contribute to the classification of a Pokemon as legendary. By leveraging the *Cross Validation Stratified* method with the *J48* model, users can make informed predictions and gain a deeper understanding of the characteristics that differentiate legendary and non-legendary Pokemon.