Problem: Classifiers based on which two attributes, if any, can predict the primary type of a Pokémon with a reasonable accuracy as opposed to the accuracy of classifiers based on all Pokémon attributes?

Chris Vajdík, s1018903

January 12, 2020

Contents

1 Abstract

This report evaluates the performance of different classifiers of Pokémons' primary types. Each of the compared classifiers is Multinomial Logistic Regression or a Decision Tree Classifier and differs in the attributes used for classifying. Every possible combination of two attributes is evaluated in terms of accuracy and its performance compared with an all attributes classifier of the corresponding type. It was found that there is little significant difference in the performance of the best two-attribute Multinomial Logistic Regression classifier and the all-attributes Multinomial Logistic Regression classifier. However, two-attribute decision trees generally performed better than the all-attribute ones.

2 Introduction

The aim of this report is to describe an implementation of Decision Tree Classifier and Multinomial Logical Regression which are used to answer the research question – classifiers based on which two attributes, if any, can predict the primary type of a Pokémon with a reasonable accuracy as opposed to the accuracy of classifiers based on all Pokémon attributes?

This is an example of a classification problem with additional efficiency comparisons. Given the relatively small dataset and high number of attributes, sufficient accuracy for real-life use should not be expected from this implementation. Nonetheless, this project illustrates the principles of classification quite well.

The general approach for answering the research question can be simplified into following steps:

- data pre-processing – select useful data and transform every data type into an integer
- get the 9-attribute classifiers
- get all the combinations of 2 pairs of attributes from the 9 attributes that can be used to predict a type and create corresponding classifiers
- compare the accuracy of the different classifiers and find the pair of attributes based on which the classifiers perform the best

Furthermore, following research sub-questions will be answered:

- Are the two 2-pair and the two 9 attributes classifiers better than random guessing? (accuracy > 5.55% (1/18*100))
- Are the accuracies of the two 2-pair classifiers and the two 9 attributes classifiers higher than always guessing the most common occurring type (water with 112 out of 800 cases)? (accuracy > 14.00% (112/800*100))
- Is the accuracy of the four classifiers higher than 60%? (so the classifiers can recognise at least the 6 most common Pokémon types)
- What is the relation of the accuracies between the two 2-pair classifiers and the two 9 attributes classifiers?
- Is there a significant difference in the performance (accuracy) of multinomial logistic regression and decision tree classifier?

There have already been many projects classifying Pokémons, three of which should be mentioned before proceeding further (see the links in Appendix 4).

'Classification Project of Pokémon Dataset' had accuracy over 95%, however the attribute to predict was binomial, so it can be safely predicted that accuracy of this implementation would be lower, as there are 18 classes to classify. This

accuracy was achieved via a Decision Tree Classifier, so the option of comparison opens up. This project uses the same Kaggle dataset.

'Pokémon type 1 classification' is the same classification problem; it aims to classify the primary type of Pokémons based on their other attributes. However, the author uses a larger database with far more attributes, so we can predict that their accuracy (>85% for Decision Tree Classifier and >89% for Multinomial Logical Regression) will be higher than the accuracy of this project.

'Pokémon Type Prediction' is the same classification problem with the same data, which uses a four layered neural network. This allows for the most interesting comparison – how well do the Multinomial Logistic Regression and Decision Tree Classifier perform in comparison with multi-layered neural networks for this data set.

3 Implementation

This section explains the implementation of the algorithms used. For the overview of data set used, see Appendix 5.

3.1 General Architecture

This implementation can be run in two modes - one is used for obtaining tweaking information for the Decision Tree Classifier and the other runs the experiment. Both of these can be run via the class Run.

Both of these modes use a class DataGetter which obtains and pre-processes the data for classification. Detailed information about obtaining the data can be found in the sub-section 3.2 Pre-processing.

The class Classifiers is used for creation of an object which contains two classifiers (Multinomial Logistic Regression and Decision Tree Classifier) for the same set of attributes. Furthermore, it contains the necessary methods to evaluate the performance of the classifiers and to produce the output to be read by the user. Tweaking class is used to optimise the performance of Decision Tree Classifier. Detailed information about classifiers and tweaking can be found in the section 3.3 Classification and its subsection 3.3.1 Tweaking.

To conclude the overview of the architecture, the class Stats (short for 'Statistics') is used to create numerous classifiers, evaluate them and pick the best ones. Details about this process can be found in the section 4 Methods.

3.2 Pre-processing

Pokémon data set needed relatively little pre-processing which is handled mainly in the class DataGetter.

Creating a new DataGetter object results in reading the whole 'Data/Pokémon.csv' file except for the first column as the identification number of the Pokémon is not useful for classification. This creates a data frame object with index column being the name of the Pokémon. Names become the index because they are not used in the classification. Classification that uses name attributes could be useful for Pokémons that have more than one card but it was not used as this would create even more imbalanced data in a small data set.

The column of the primary types of Pokémons then becomes the y data and gets translated from strings to integers. This is the attribute to be predicted. The rest of the columns, except for the index column, becomes the X data. Secondary type of the Pokémons gets translated to integers as well along with the legendarity status.

The pre-processing process ends with normalisation of the data before it is used by classifiers.

3.3 Classification

The classification processes are handled by the class Classifiers. Logistic Regression and Decision Tree Classifier from the Sklearn library are used. The process of obtaining the parameters for the Decision Tree Classifier is handled by a separate class Tweaking and is described in the subsection below.

An object of the type Classifiers contains a Logistic Regression object and a Decision Tree Classifier object along with their input data, training and testing processes and resulting accuracies.

This class can return a tuple of accuracies - the training Logistic Regression accuracy, the testing Logistic Regression accuracy, the training Decision Tree Classifier accuracy and the testing Decision Tree Accuracy. These are then used to pick the best classifiers. This process is described in Section 4.2 Obtaining Statistics.

The class Classifiers can also produce an informative output for the user, an example of which can be found in Appendix 8.2 Accuracy results. The output contains the names of attributes that the classifiers use and statistics (average accuracy in train and test sets and the set of all observed accuracies).

3.3.1 Tweaking

In order to obtain the most accurate results, the Decision Tree Classifier needs well-chosen parameters for at least the maximal depth of the tree. The value of this parameter along with the values of maximum number of features to be considered, minimum number of samples in a leaf node and minimum number of samples needed to split were determined in the class Tweaking.

Tweaking runs the train/test sequences for the same data set using different values of attributes and notes the accuracy for each value. Afterwards, it produces graphs which can be seen in Appendix 8.1 Tweaking Results. It is needed to manually configure the Decision Tree Classifier in the class Classifiers, because doing this automatically would add unnecessarily many attributes when initialising the class Classifiers. Choosing and using the optimal parameters automatically could be useful if there were more data sets to work with or more classifiers to use but it is not worth the additional complexity (for humans to understand the code) to add such a feature when it would only have been used once.

As it can be seen in graphs a), c) and d), the most of the parameters have to be specified to prevent over-fitting (train sets being more accurate than test sets). In the graph b), it is clear that a certain degree of overfitting cannot be prevented (the train set accuracy always being 1 no matter how many features are considered), which might be due to the data set being relatively small while having relatively many classes.

The values of the parameters were chosen depending on the relation of test and train accuracies. Ideally, both should be as high as they can get. However,

the testing accuracy is more important than the training accuracy since too high training accuracy usually indicates overfitting. This means that the values of the parameters where the test accuracy has its peak were chosen.

4 Methods

In order to choose the best classifiers, a way to obtain and compare the performance of a classifier had to be created and a way of automatic creation of a relatively high number of classifiers had to be developed. The first subsection of this section deals with the way to obtain and compare classifiers' performance and the second is about the former mentioned problem.

4.1 Performance of a Classifier

There are many classification metrics that can be used in classification problems. Which of these metrics is chosen depends on the data set and the problem characteristics.

Since the data set of this problem has multiple classes and the aim of the project is to compare the classifiers and not necessarily find the best classifier for the data set, it was decided that the metrics used should be simple so as not to draw attention away from the research questions.

The classes are relatively balanced with the most populated class taking up 14% of the data entries. There is no goal that would require an especially strong confidence in the model which would indicate a need for specificity measurement or other more complicated measurements. Because of these two reasons, the performance metrics chosen is accuracy.

An object of the class Classifiers contains two classifiers, each of which is trained and tested on different fractions of the data set. Therefore, this object contains 4 information about the accuracy (the training Logistic Regression accuracy, the testing Logistic Regression accuracy, the training Decision Tree Classifier accuracy and the testing Decision Tree Accuracy). To compare two objects of the Classifiers class, the best approach is to compare test accuracy of one Logistic Regression classifier with the other and do the same with the Decision Tree Classifier. Depending on the goal, one or the other classifier type will yield more important comparison (e.g. to choose the better Decision Tree Classifier, one does not have to compare Logistic Regression classifiers at all). Comparing training results should not be prioritised as this would rank potentially overfitting models higher than the non-overfitting ones.

4.2 Obtaining Statistics

To get the best two-attributes classifiers, it is required to create a classifier for each combination of 2 attributes out of the nine possible attributes to use, obtain accuracy of every one of them and compare them (this process must be done both for Logistic Regression and Decision Tree Classifier). The class Stats both obtains the statistic of every possible classifier and compares them to choose the best one.

Using the itertools library, an iterable object of all the possible pair of numbers 0 to 8 is created. For every one of these pairs, a Classifiers object is created and its accuracies are obtained. To obtain the best classifier of one of the two

classifier types, the array of accuracies is sorted by the test performance of the specified classifier.

5 Results

The Logistic Regression classifier with 9 attributes had an average test accuracy of 19.87%. For the train accuracy, it was 23.69%, which is 3.82% higher than the test accuracy. The Decision Tree Classifier with 9 attributes had an average test accuracy of 13.99%. For the train accuracy, it was 16.25%, which is 2.26% higher than the test accuracy. The Logistic Regression is 1.42 times more accurate on testing data than the Decision Tree Classifier and 1.45 times more accurate on training data.

The best two-attribute Logistic Regression classifier and Decision Tree Classifier both use attributes HP (health points) and Sp. Atk (special attack). For Logistic Regression, the average test accuracy was 19.75% while the train accuracy was 19.37%. For Decision Tree Classifier, the average test accuracy was 17.24% while the train accuracy was 17.44%. The Logistic Regression is 1.15 times more accurate on testing data than the Decision Tree Classifier and 1.11 times more accurate on training data.

The average test accuracy of a nine-attribute Logistic Regression was slightly higher (1.006 times higher) than the average test accuracy of the best two-attribute Logistic Regression, while for the train data, it was 1.22 times higher.

The opposite was true for the Decision Tree Classifiers, where the average test accuracy of the best two-attribute classifier was 1.23 times higher than the average test accuracy of the nine-attribute classifier. For the train accuracy, the two-attribute classifier had 1.07 times higher accuracy than the nine-attribute classifier.

This shows that every type and sub-type of the classifiers is better than random guessing and all but the 9 attribute Decision Tree Classifier are better than always guessing the most commonly occurring type. However, none of the classifiers had the accuracy higher than 60%, so they do not recognise the 6 most common Pokémon types.

It seems that logistic regression performs better when the attributes are not restricted, since the nine-attribute classifier had a higher accuracy, while the opposite holds for the Decision Tree Classifier, which probably stems from the characteristics of the two methods. The more attributes the Decision Tree Classifier considers, the lower its accuracy due to overfitting. However, the Multinomial Logistic Regression works with the probabilities (assigns a threshold value for each class on each attribute), so using more attributes than Decision Tree Classifier does not constitute a problem.

For this data set, it is clear that the Multinomial Logistic Regression performs slightly better. This might be because unlike Decision Tree Classifier, it does not use any sort of hierarchy - it seems that in the data set used, it is hard to rank the attributes by importance (the power with which they affect the type of a Pokémon).

6 Conclusion

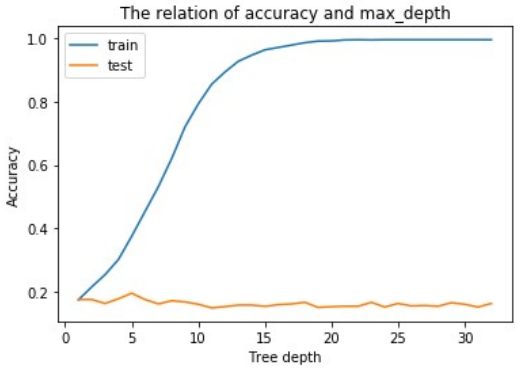
To conclude, every classifier created performed better than random guessing, three out of four performed better than always guessing the most populated class but no classifier had accuracy over 60%.

This relatively poor performance (as compared to similar projects) may be due to the nature of the data set (when comparing with the project Pokémon type 1 classification), due to the types of classifiers (when comparing with the project Pokémon Type Prediction) or due to characteristics of the problem (when comparing with Classification Project of Pokémon Dataset). It is clear that this data set in relation to the problem being solved did not contain enough data for the accuracy to be good enough for real world application because even the approach using neural network did not generally have accuracy higher than 60%. The data set, however, contains enough information for an over-90% accurate binomial classifier. It appears than using neural networks based classifiers would yield significantly better results.

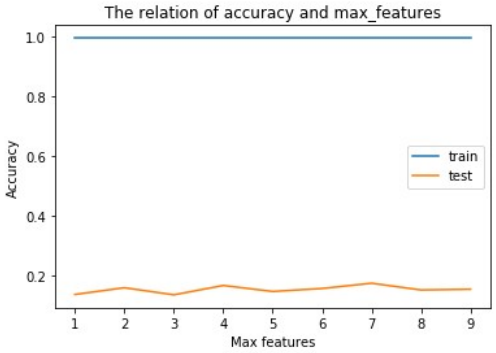
Logistic Regression classifier performed slightly better than the Decision Tree Classifier. However, there was a significant difference only when using all nine attributes. When comparing all-attribute classifiers with the best two-attribute classifiers, it was observed that Logistic Regression performed worse with less attributes but the opposite holds for the Decision Tree Classifier.

7 Appendices

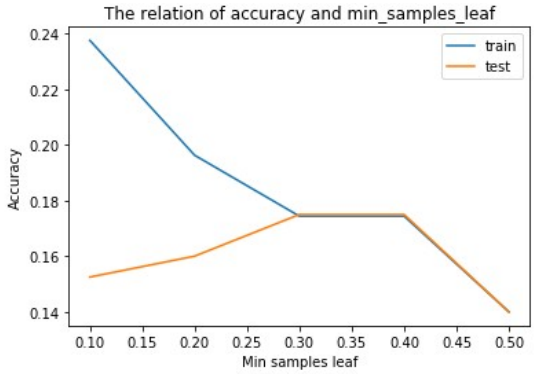
7.1 Tweaking Results



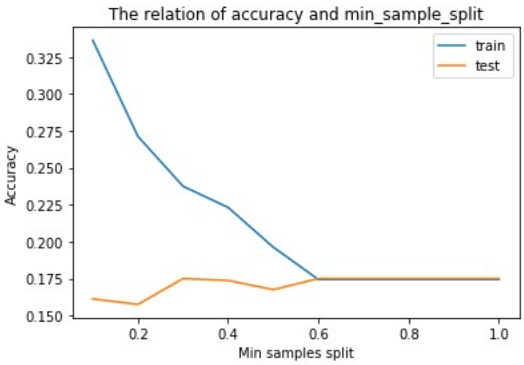
(a) Maximal depth of the decision tree



(b) Maximum number of features considered



(c) Minimum number of samples in a leaf node



(d) Minimum number of samples needed to split

Figure 1: Plots of data obtained while tweaking the Decision Tree Classifier

7.2 Accuracy results

```
------

CLASSIFIERS WITH 9 ATTRIBUTES

------


Classifiers with attributes:  ['Type 2', 'Total', 'HP', 'Attack', 'Defense', 'Sp. Atk', 'Sp. Def', 'Speed', 'Generation', 'Legendary']

---
LR
---

Accuracy in train set:  [0.22889305816135083, 0.23076923076923078, 0.250936329588015]
AVG accuracy in train set:  0.23686620617286555
Accuracy in test set:  [0.20973782771535582, 0.21722846441947566, 0.16917293233082706]
AVG Accuracy in test set:  0.1987130748218862

---
DT
---

Accuracy in train set:  [0.16135084427767354, 0.14634146341463414, 0.1797752808988764]
AVG accuracy in train set:  0.16248919619706137
Accuracy in test set:  [0.20973782771535582, 0.12359550561797752, 0.08646616541353383]
AVG Accuracy in test set:  0.13993316624895571
```

(a) The pair of 9-attribute classifiers

```
Classifiers with attributes:  ['HP', 'Sp. Atk']

---
LR
---

Accuracy in train set:  [0.17448405253283303, 0.1951219512195122, 0.21161048689138576]
AVG accuracy in train set:  0.19373883021457697
Accuracy in test set:  [0.23220973782771537, 0.20224719101123595, 0.15789473684210525]
AVG Accuracy in test set:  0.19745055522701885

---
DT
---

Accuracy in train set:  [0.16135084427767354, 0.1669793621013133, 0.1947565543071161]
AVG accuracy in train set:  0.1743622535620343
Accuracy in test set:  [0.20973782771535582, 0.1797752808988764, 0.12781954887218044]
AVG Accuracy in test set:  0.17244421916213756
```

(b) The most accurate Multinomial Logistic Regression and Decision Tree Classifier with 2 attributes

Figure 2: Outputs of program regarding the statistics of specific classifiers

7.3 Source Code

The source code for this project can be found here.

7.4 Related Work

The related work that was mentioned in this project can be found on these links:

- Classification Project of Pokémon Dataset - the same data set, a different problem,
- Pokémon type 1 classification - a different data set, the same problem with a different approach),
- Pokémon Type Prediction - the same data set, the same problem with a different approach.

7.5 Data Set Used

The data set used in this problem can be found here.

It contains 721 unique Pokémons and 800 total entries with the following attributes:

- number - the number of the Pokémon (remains the same for the same Pokémon if it has more cards)
- name - a string
- primary type - a string that informs about the type of the Pokémon, 18 options
- secondary type - can be none
- evaluation of the total power of the Pokémon - integer
- health points (HP) - integer
- attack power - integer
- defence power - integer
- special attack power - integer
- special defence power - integer
- speed - integer
- generation - integer
- legendarity status - Boolean