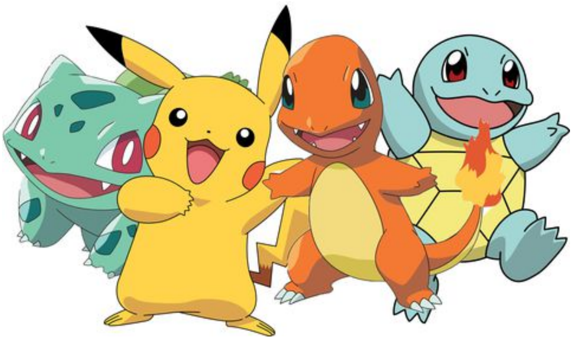
Technical University of Denmark



02540: Introduction to Machine Learning and Data Mining
Project Two:

Erik Gylling, s173896
Christian Hinge, s174173

April 9th, 2019

Contents

Before continuing with the assignment, we will quickly mention some of the data-transformations that are in use. Categorical attributes have been K-encoded and the two attributes **Height** and textbfWeight have been log-transformed. Furthermore, all data used is normalized with consideration to the the K-encoding. The attribute **Total** has been removed, as this is simply a sum of three other attributes.

Regression goals

Attack is a general measure of a Pokémon's strength and how the creature will perform in battle. We found this continuous ratio attribute interesting and decided try and predict it via regression models.

Regression a

For now, all attributes will be used in the regression models. The general form of the linear model is:

$$Attack = \mathbf{W}^T \cdot \mathbf{X} \tag{1}$$

Where $\mathbf{W}$ is a vector with the weight of each attribute and a bias, w_0. $\mathbf{X}$ is a vector with all attribute-values for a given observation and a constant term x_0 which is equal to 1.
The next step is to estimate the generalization error for different λ-values using cross-validation. Initially, we will use a wide interval $[10^{-5}; 10^{8}]$.

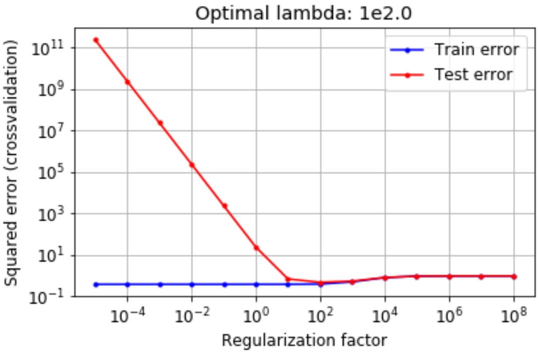


Figure 1: Test Error and Training Error as a function of λ

The optimal λ is found at the point where the model generalizes well, i.e.: where the variance-bias trade off yields a model that is not too complex nor too simple. This sweet spot is found where the test error is at its minimum as the test error is a good approximation of the generalization error. According to figure 1, the optimal λ is roughly equal to 10 to 100

A Pokémon's attack power can now be estimated by taking the dot product of the Pokémons attributes (excluding the Attack power) and the calculated weights in the linear model. The weights are found by solving the regularized normal equation[1] with λ set to the optimal value (100 in this case). Depending on these weights, some attributes will have more influence on Attack power than others. **HP** for instance has a positive and large weight, so if the Pokémon's **HP** attribute is large, the model will predict a larger **Attack** value.

[1] Machine Learning book, page. 214, eq. 13.4

A Pokémon will have a significant *lower* Attack value if the Pokémon is **Pink**, has a high **Catch_rate** or has a **Head-only** body-type (see figure 8b). The Pokémon will have a signigicant *higher* Attack Value if it is of type **Dark**, has a **megaEvolution**, high **SP.Atk**, high **Speed**, high **Pr.Male**, or is **Legendary**.
The Pokémon type dark is one of the scarier Pokémon-types in the game, while pink and female Pokémons usually are cute. It seems that Pokémons that are more interesting for boys have higher attack values. The Pokémon franchise targets mainly an audience of males which probably explains the gender-discrimination.
It makes sense that Pokémons with a high **Catch_Rate** have a lower attack value, since stronger Pokémons are normally harder to catch. Pokémons of type "head-only" body-type are normally using attacks that scales with **SP.Atk** instead of **Attack**, which explains the correlation.
megaEvolution, and **isLegendary** are true if the Pokémon is rare and exquisite. Thus it makes sense that these attributes result in high Attack power as well.

Regression b

Artifical Neural Network

For the artifical neural network, we initially used the number of neurons in the hidden layer as the complexity controlling parameter. We tested for values between 1 and 10 inclusive, and found that the inner cross validation would choose 2 as the optimal number of hidden neurons 90% of the time (the other 10% being 1 and 3). Higher numbers of hidden neurons gave tiny training errors but large test errors, which is a clear indication of overfitting.

However, we quickly became aware of another parameter of the network which had far greater influence on the test error; the number of training iterations. When we were ready to produce plots and data for the assignment, we turned up the maximum number of training iterations of the neural network from 2,000 to 50,000 thinking that longer training time would result in a better ANN which would yield better predictions. This assumption was quickly shown to be incorrect as the network that had done 50,000 training iterations overfitted the data.
We did som research, and found that "Early stopping"[2] is a valid and used regularization technique to avoid overfitting iterative machine learning models such as neural networks, which utilize iterative gradient descent. As a result, we changed the controlling parameter from hidden neurons to the number of max training iterations. After doing a few trial runs, we narrowed down the interval with the optimal parameter value to [100,1500] (which includes some padding for extra measures). Figure 2 clearly shows the parameter's impact on generalization error. Parameter values are withdrawn from the interval in steps of 100, i.e: 100, 200, ... , 1500 (see figure 3). We tried out a different number of hidden neurons to ensure that the old parameter wasn't too correlated with the new, which seems to be true. Optimally, 3-layer cross-validation should be used to optimize both parameters. Our computers did however not allow for this. Furthermore, we decided to let PyTorch repeat each training session three times to avoid the consequences of unfortunately placed local minimas in the gradient descent.

Regularized Linear Model

For the regularized linear model we rewrote and replaced almost all functions in the toolbox to ensure that the same data-splits were used across all three models. Other than that, based on the results in Regression part a we narrowed down the optimal λ-interval to [1,200] which we tested in

[2] https://en.wikipedia.org/wiki/Early_stopping

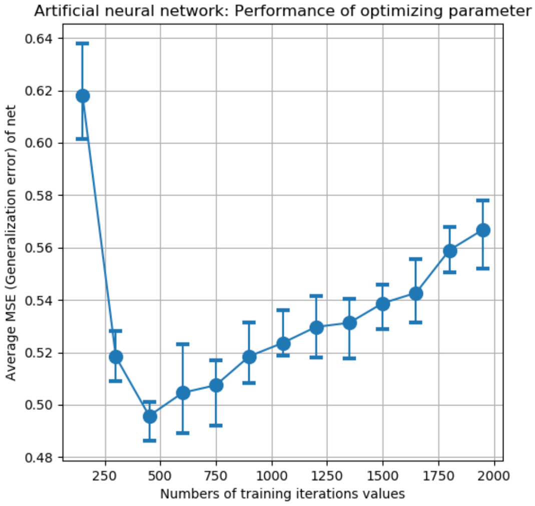


Figure 2: Average generalization error of different number of iterations. The error bars represent the IQR

steps[3]. of 10

Results

For the generalization error, we decided to go with the MSE. The final generalization errors are calculated as the weighted average of the MSE's in table 1 as prescribed in algorithm 6. The generalization errors for the three models are:

- **Artifical neural network:** 0.5107
- **Linear model with regularization:** 0.4743
- **Baseline:** 1.003

Outer fold	ANN		Linear regression		Baseline
i	h_i^*	E_i^{test}	λ_i^*	E_i^{test}	E_i^{test}
1	400	0.584	40	0.651	1.158
2	600	0.431	50	0.446	0.825
3	400	0.395	30	0.371	0.749
4	500	0.405	40	0.421	1.009
5	500	0.390	30	0.418	0.984
6	500	0.442	30	0.431	0.849
7	500	0.720	30	0.694	1.195
8	400	0.311	40	0.314	0.850
9	600	0.589	30	0.571	1.101
10	600	0.559	30	0.527	1.313

Table 1: Table of estimated generalization errors for each outer fold

[3] We did not test for 0, as some splits yielded columns where all elements were identical. This resulted in a singular matrix which could not be used in the normal equation

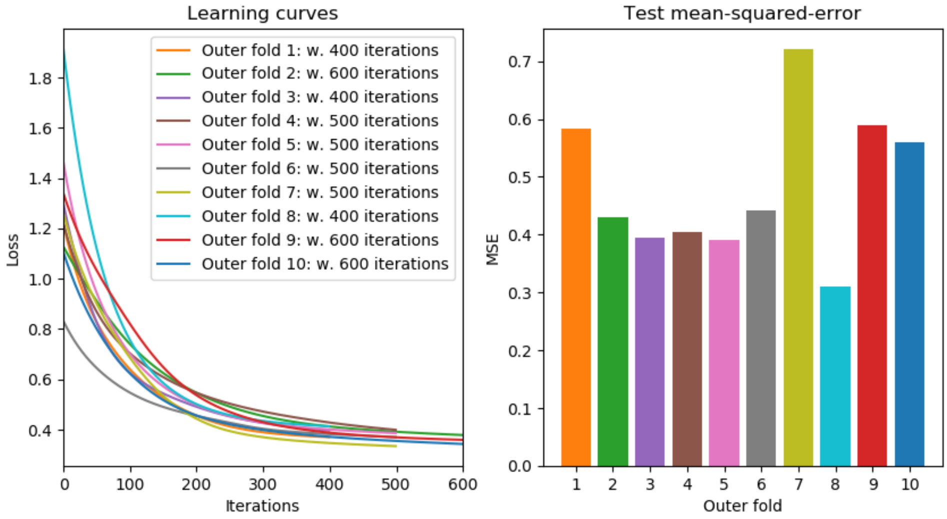


Figure 3: Training and generalization of the neural network

Since our data is normalized it makes sense that the MSE for the Basline is just above 1 as baseline calculates and predicts only based on the training part of the dataset, of which the mean in most cases will deviate a tiny amount from 1.
As for the linear model and the ANN. They both seem to perform okay. True versus estimated values have been plotted both for the best ANN (Figure 4a) and linear model (Figure 4b). By "best models" we refer to the models trained and tested in the *outer* cross validation folds.

Our null-hypothesis will now be put to the test:

$$H_0 : \text{ All three regression models generalize equally well} \tag{2}$$

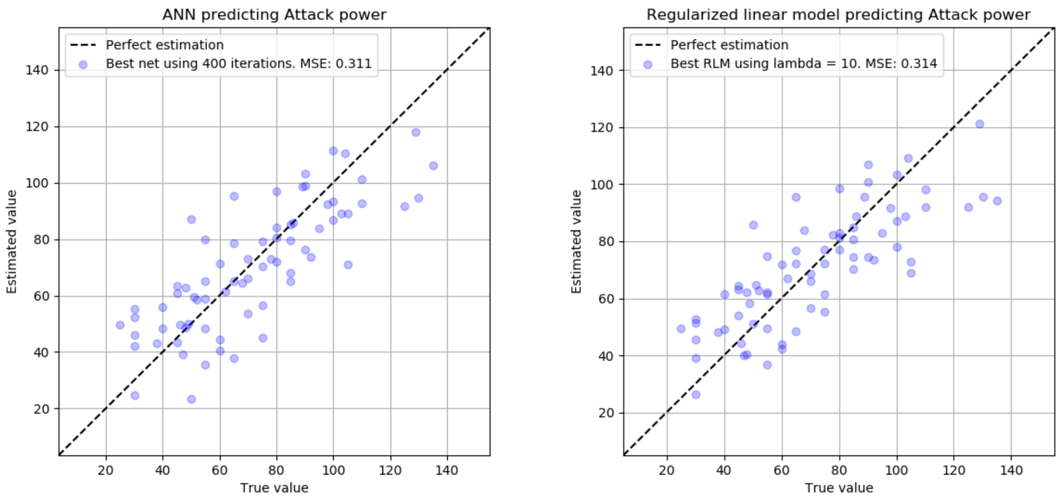
Since the splits were identical for the three models, we can do pairwise t-tests and confidence or prediction intervals on the error-differences. The confidence interval is calculated as so:

$$\tilde{x} \pm t_{1-\alpha/2} \cdot \frac{s}{\sqrt{n}} \qquad \text{,where t follows a t-distribution with 9 degrees of freedom} \tag{3}$$

And yields the following confidence intervals for the difference in model generalizations:

- **Baseline - RLM**: [-0.600, -0.438], p-value $< 3 \cdot 10^{-7}$
- **Baseline - ANN**: [-0.602, -0.439], p-value $< 3 \cdot 10^{-7}$
- **ANN - LM**: [-0.0225, 0.0192], p-value $= 0.9$

The 95% confidence intervals and p-values clearly show that we have very strong evidence against the null-hypothesis as both the ANN and RLM out-perform the baseline. Whether the ANN or RLM is best however, is uncertain as 0 is included in confidence interval. This is indeed also what we observed in other trial runs.
Obviously, the baseline cannot be recommended. Both the ANN and RLM show promising results. The RLM is way faster to train and is thus the winner for now.



(a) True versus predicted values for the best artifical neural network in the outer fold

(b) True versus predicted values for the best regularized linear model in the outer fold

Figure 4

Classification

For classification, we decided to predict whether or not a Pokémon is legendary. The artificial neural network proved itself to be useful in regression, but the results were not better than regularized linear regression. Therefore, we decided to choose Classification Tree as our ”method 2”. This binary classification problem will hence be solved using the following three methods:

- Baseline
- Logistic Regression
- Classification Tree

The results are summed up in table 2. All attributes, except from the one we are predicting, will be used in the models.

2-Layer Cross Validation

2-layer cross validation is used to compare the performance of the three models. The error rate, that will be used as an error measure is:

$$E = \frac{\{\text{Number of misclassified observations}\}}{N^{test}}$$

Baseline

The estimated generalization error of the baseline is 6.38% which one might think is quite good. This, however, is simply due to the uneven distribution of legendary/non-legendary Pokémons. Obviously this is not a great classifier as no Pokémon is ever classified as legendary.
This is shown in figure 6b.

Logistic Regression

For the logistic regression regularization parameter, we did some trial runs to narrow down the interval of possible λ values to $[10^{-6}; 10^{2}]$. The results of the final model clearly show that there's an optimal value in this range at around $\lambda = 1$ (figure 7b)
This is evident in table 2 where all the λ were chosen to be either 1.668 or 0.215 in all 10 outer-folds.

The overall generalization error for the logistic regression is estimated to 1.23% which is a significant improvement compared to the baseline. In several outer folds the logistic regression perfectly classified all Pokémons (figure 6c).
Which of the attributes the logistic regression employs to achieve such an accuracy will be discussed later when we compare it with the linear regression.

Classification Tree

The criterion used for the classification tree is gini. The depth parameter used for classification trees is found by a trial over nineteen parameters in the interval [2;21]. The optimal depth is the depth with lowest test-error. In table 2 it can been seen that the depth of 2 is chosen many times. That's a low depth especially when there are 46 legendary Pokémons out of a total of 721 Pokémons. So legendary Pokémons will probably have something in common.
Most of the legendary Pokémons are really hard to catch. So the **Catch Rate** is low. So the first question to ask is *"Is the catch rate under a certain value"*. After this question the legendary Pokémons is almost seperated from the rest of the Pokémons. Another attribute that all legendary Pokémons have in common is the egg type **Undiscoverd**. Legendary Pokémons usually can't lay eggs so their egg type is **Undiscoverd**. And apparently the legendary Pokémons that is easier to catch has a high value of **SP.Atk**. So it wasn't that difficult to seperate legendary Pokémons from not legendary Pokémons. The classification tree is showed on figure 5.

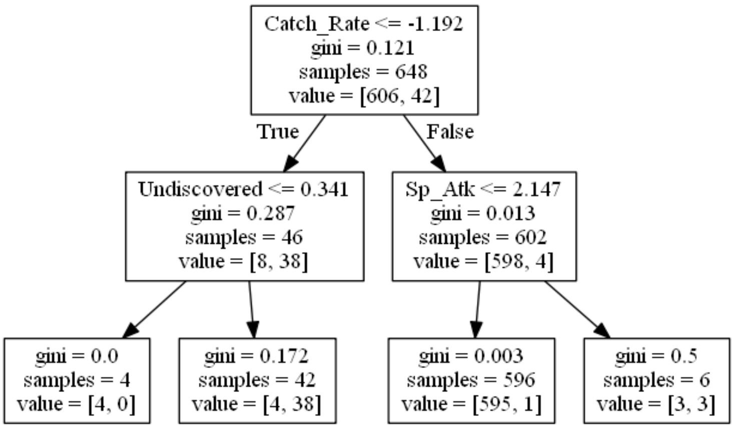
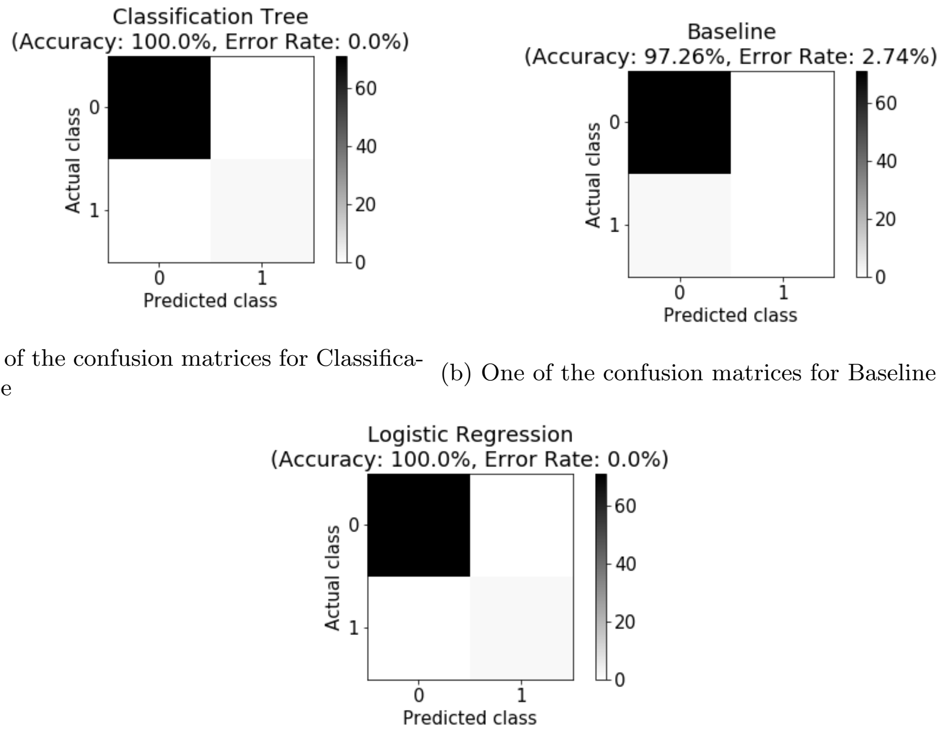


Figure 5: The classification tree with depth of 2

On figure 6a one of the confusion matrice for the classification tree is shown. It can be seen that the accuracy is perfect and it succeeded to classify the legendary Pokémon(s) correctly with tree structure of figure 5.

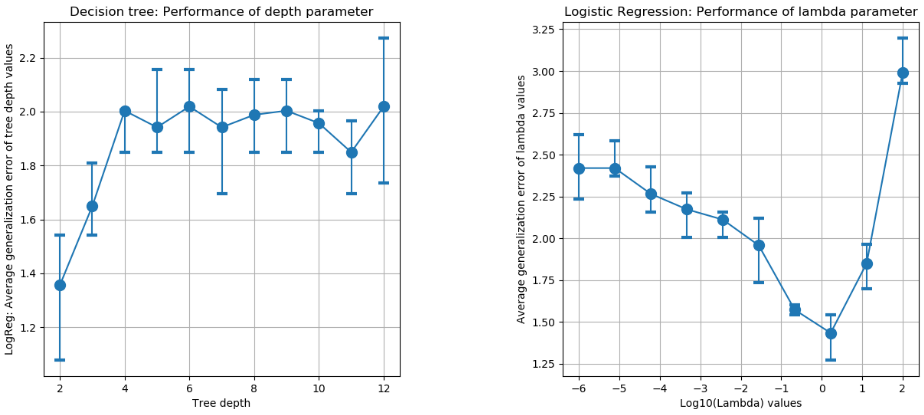


(a) One of the confusion matrices for Classification Tree

(b) One of the confusion matrices for Baseline

(c) One of the confusion matrices for Logistic Regression

Figure 6



(a) Average performance of decision tree for different values of the tree depth. The error-bars indicate the IQR

(b) Average performance of logistic regression for different values of λ. The error-bars indicate the IQR

Figure 7

Results

Outer fold	C. Tree		Logistic Regression		Baseline
i	h_i^*	E_i^{test}	λ_i^*	E_i^{test}	E_i^{test}
1	2	1.370%	1.668	0.000%	8.219%
2	2	0.000%	0.215	1.389%	5.556%
3	2	1.389%	1.668	1.389%	5.556%
4	2	1.389%	1.668	0.000%	5.556%
5	2	2.778%	0.215	2.778%	6.944%
6	2	1.389%	1.668	1.389%	5.556%
7	3	1.389%	0.215	2.778%	6.944%
8	2	1.389%	0.215	0.000%	8.333%
9	2	0.000%	1.668	1.389%	5.556%
10	2	0.000%	0.215	0.000%	5.556%

Table 2: Table showing the optimal λ's , depths, and the test error for each outer fold in the cross validation.

Table 2 shows the the generalization errors measured in percent. The overall weighted generalization errors for the three models are:

- **Baseline**: 6.38%
- **Logistic regression**: 1.23%
- **Classification tree**: 1.23%

Like in the regression section, we can use pair-wise t-tests and confidence intervals to test the null-hypothesis:

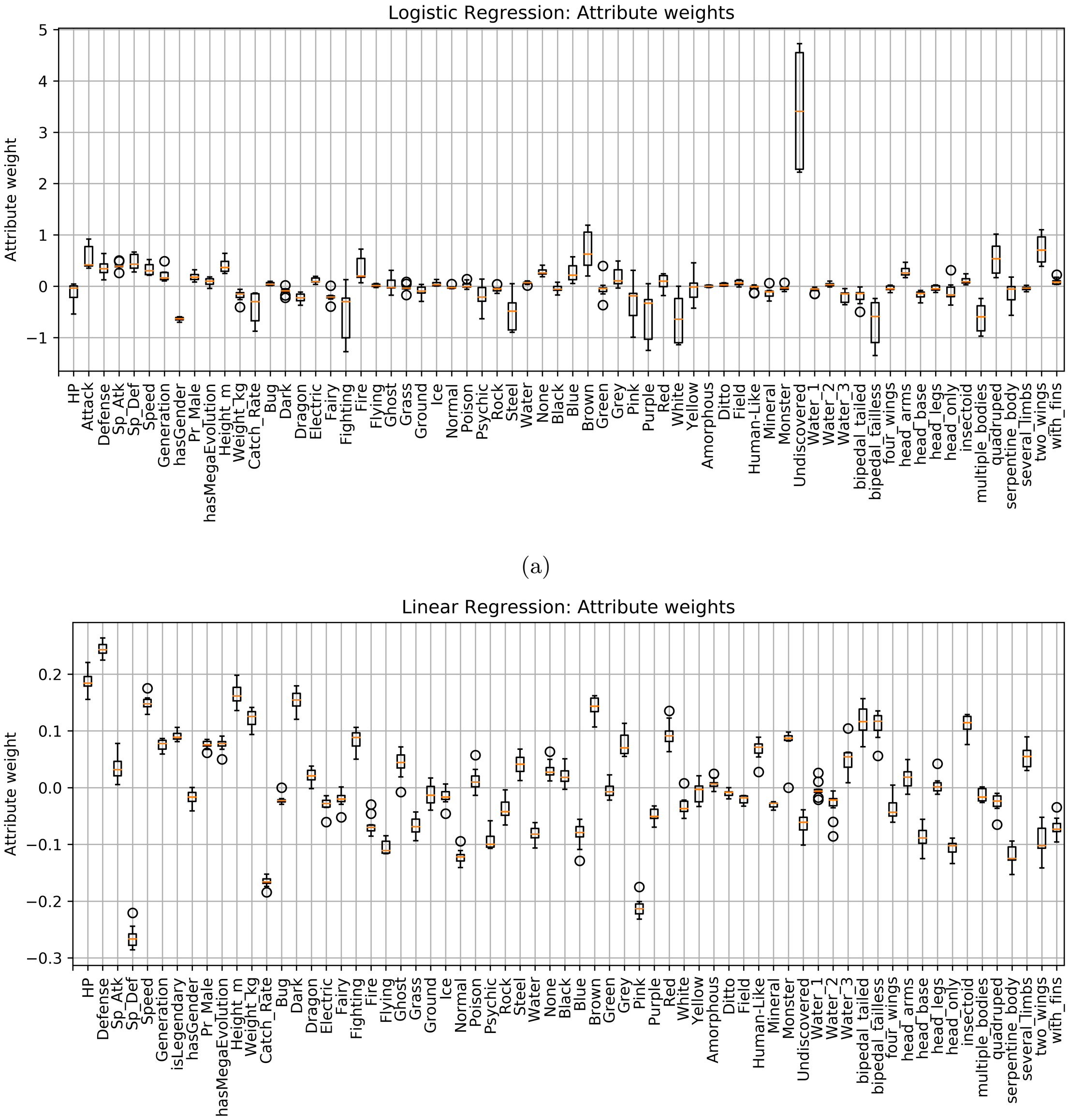
- **Baseline - CT**: [-6.001%, -4.535%], p-value $< 9 \cdot 10^{-8}$
- **Baseline - LR**: [-6.410%, -4.123%], p-value $< 5 \cdot 10^{-8}$
- **CT- LM**: [-0.766%, 0.770%], p-value = 1.00

The 95% confidence intervals and p-values clearly show that we have very strong evidence against the null-hypothesis as both the LR and CT out-perform the baseline. Whether the LR or CT is best however, is uncertain as 0 is included in confidence interval.

Comparison of Logistic Regression and Linear Regression

The linear and logistic regression models discussed previously were used for regression and classification respectively. Nevertheless, it's interesting to see if the models picked out any of the same attribute patterns.

The logistic regression model can be visualized as a raised plateau with a soft, but quick, transition down to a floor area. This ensures that the majority of the vector-space belongs to either the plaeteu (1/true) or floor (0/false) and that all function values will be in the range [1 ; 0]. When the optimal parameter weights for the model have been found, predictions can be performed by clamping the returned function values to either 1 or 0 - whichever one is the closest, i.e: 0.7 is clamped to 1 while 0.45 is clamped to 0.


Logistic Regression: Attribute weights
Attribute weight
5
4
3
2
1
0
−1
HP
Attack
Defense
Sp_Atk
Sp_Def
Speed
Generation
hasGender
Pr_Male
hasMegaEvolution
Height_m
Weight_kg
Catch_Rate
Bug
Dark
Dragon
Electric
Fairy
Fighting
Fire
Flying
Ghost
Grass
Ground
Ice
Normal
Poison
Psychic
Rock
Steel
Water
None
Black
Blue
Brown
Green
Grey
Pink
Purple
Red
White
Yellow
Amorphous
Ditto
Field
Human-Like
Mineral
Monster
Undiscovered
Water_1
Water_2
Water_3
bipedal_tailed
bipedal_tailless
four_wings
head_arms
head_base
head_legs
head_only
insectoid
multiple_bodies
quadruped
serpentine_body
several_limbs
two_wings
with_fins
Linear Regression: Attribute weights
Attribute weight
0.2
0.1
0.0
−0.1
−0.2
−0.3
HP
Defense
Sp_Atk
Sp_Def
Speed
Generation
isLegendary
hasGender
Pr_Male
hasMegaEvolution
Height_m
Weight_kg
Catch_Rate
Bug
Dark
Dragon
Electric
Fairy
Fighting
Fire
Flying
Ghost
Grass
Ground
Ice
Normal
Poison
Psychic
Rock
Steel
Water
None
Black
Blue
Brown
Green
Grey
Pink
Purple
Red
White
Yellow
Amorphous
Ditto
Field
Human-Like
Mineral
Monster
Undiscovered
Water_1
Water_2
Water_3
bipedal_tailed
bipedal_tailless
four_wings
head_arms
head_base
head_legs
head_only
insectoid
multiple_bodies
quadruped
serpentine_body
several_limbs
two_wings
with_fins

(a)

(b)

Figure 8

As for importance of features, the logistic regression model relies heavily on the **Undiscovered** attribute which is a K-encoded attribute originating from the attribute **Egg-group** (see figure 8a). Legendary Pokémons are rare and mysterious which explains the relation to **Undiscovered**. Another sign of this is that Legendary Pokémons rarely have known genders. Apart from that, Legendary Pokémons are usually strong, which is shown by the attributes **Attack**, **Defense**, **Speed-Attack**, **Speed-Defense**, and **Speed**. These attributes are correlated, which is also true for the linear regression (see figure 8b). This is however as far as the similarities between the two models go.

Discussion

During this assignment we learned to appreciate the importance of cross-validation when analyzing performance. At the same time, we became aware of how easy and common over-fitting data really is. There are tons of parameters to tune, and machine learning does not always have best and definite answers to a problem.

For this assignment, we decided not to use principle components. It is a trade-off between making the model predict better and being able to conclude on the model weights. If we were to revisit this subject, we would definitely consider the possibility of using PCA, and maybe forward/backwards selection of attributes. For the classification part, bagging or boosting could even out the **isLegendary** attribute. Looking back at the classification results (table 2), it seems that the the generalization errors for the Logistic Regression and Decision tree are fairly independent. Thus ensemble methods could be implemented to strengthen the classifier.

Previous Work

This dataset has been used by Asier López Zorrilla in January 2017[4] to predict **Catch_Rate** with 4 linear models. His best model was the third model with $R^2 = 0.5466$.

Asier López Zorrilla has also done a Linear Discriminant Analysis to predict if a Pokémon is legendary or not. He used two PCA's as input an his model predicted 29 pokémons wrong out of 721 which is 4%.

His results are similar to our results, but we made two models that were way better at classifying legendary Pokémons than his (1.23% compared to 4%). We found out that legendary Pokémons have a high **Catch_Rate** and an **Egg_group** type **Undiscovered**. These two attributes might not have been sufficiently expressed in the two PCA's he used.

Appendix

Workload

Section	Erik Gylling	Christian Hinge
Regression a	70%	30%
Regression b	30%	70%
Classification	60%	40%
Discussion	50%	50%

[4] Zorilla, A., *Statistical Analysis of Pokémon*, 2017.
The pdf can be found on www.kaggle.com/alopez247/pokemon (visted: 8th of april 2019)