ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
UNIVERSITY OF CRETE

Machine Learning Methods: Application to Abalone snails age prediction

Panagiotis Anastasakis
Aitor Beñaran
Eleni Dretaki

Department of Mathematics & Applied Mathematics
University of Crete
Heraklion, Greece

July 3, 2022

Contents

1 Introduction

Abalones snails are a marine species found in cold coastal waters around the world. Humans have been harvesting abalones for centuries as a source of food, while their shells are often used as decorative items. Unfortunately, nowadays the species is under the threat of extinction due to overfishing and the acidification of the oceans from carbon dioxide.

Extensive research has been done to better understand the species and to find ways to preserve their population, one of the most important aspects of which is determining the age of the abalones. In general, this task is very complex and time consuming since it requires cutting the shell through the cone, staining it, and counting the number of rings through a microscope. Then, the age is computed by the following simple equation:

$$\text{age} \;=\; \text{number of rings} \;+\; 1.5 \tag{1}$$

A machine learning model that predicts the age of the abalones, using measures that are easier and faster to obtain, can be a great tool for scientists. That model will assist the efforts of studying the abalones and ensuring their conservation. In this project we are predicting the age of the abalone snails based on various measurements like sex, length, diameter, etc.

In order to make the predictions, we are going to focus on regression and classification algorithms. We will present all the methods that we are going to use, specifying what type of algorithms are they (regression or classification) and introducing the mathematical basis of each of them. Moreover, we will implement two types of classification, one concerning each ring value as a class, and another considering the "young" and "old" abalone classes (binary classification).

We have obtained the abalone dataset from Kaggle [1] and most of the methods have been considered from the book An Introduction to Statistical Learning [2]. Other methods have been considered from different books and webpages (see bibliography). Finally, in order to implement those models, scikit-learn webpage [3] has been used.

2 Methodology - Techniques

The abalone dataset that was used in this project was sourced from the UC Irvine Machine Learning Repository published in 1995. Each row (4176 in total) in the dataset represents the attributes and physical measurements of Abalones including sex, length, diameter, height, whole weight, shucked weight, viscera weight, shell weight and number of rings. The number of rings were counted manually using a microscope by the researchers. Note that the sex variable in the dataset includes 3 categories: male, female and infant. This is rather curious since the age of the abalone is binary (male / female) and infant is not really considered a sex of abalone but instead is a reference to its age.

The dataset has already its missing values removed and the range of the continuous values have been scaled for use with Artificial Neural Networks (ANN), by dividing them by 200. For the purposes of this project, we further normalized the data using the standard normal distribution. For the regression part, we removed the ring column and replaced it with the age based on (1). Moreover, for the young / old classification, we separated the abalone dataset into those two classes, based on the number of their rings. Specifically, we considered an abalone young (label 0) when having 10 rings or less and old (label 1) otherwise.

	sex	length	diameter	height	whole weight	shucked weight	viscera weight	shell weight	age
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	16.5
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	8.5
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	10.5
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	11.5
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	8.5
...	...	...	...	...	...	...	...	...	...
4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	12.5
4173	M	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	11.5
4174	M	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	10.5
4175	F	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	11.5
4176	M	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	13.5

Figure 1: Abalone dataset for regression algorithms

Before making any predictions, we performed some simple data visualization in order to have a deeper understanding of how the biological measurements are correlated and how they affect the age of the abalone snails. For that, we chose a variety of statistical plots offered from the MatPlotLib and Seaborn libraries in Python, specifically using the following tools:

- Histograms of all float type features, as well as the age
- Boxplots of the features
- Heatmap depicting the correlation of the features
- Scatter plots, using as variables each feature and the age

The first part of the project is to predict the age of the Abalones. Our initial approach to achieve this was to implement some regression algorithms. After research, we decided to use the following machine learning algorithms for regression:

- Linear Regression
- Ridge Regression
- Lasso Regression
- Decision Tree
- Support Vector Regression (using both the Linear and the Radial Basis Function kernels)
- K-nearest neighbors

In the dataset, there are 29 different results adquired from counting the number of rings (every integer from 1 to 29), meaning that the age of the Abalone will belong in one of the 29 different classes, given that we exclude the rare case of the age being greater than 30. Therefore, another way to view this problem is through multiclassification with 29 classes. For this approach we implemented the following machine learning algorithms for classification:

- Linear Discriminant Analysis
- Support Vector Machine, One-Versus-One
- Support Vector Machine, One-Versus-All
- Decision Tree
- K-nearest neighbors

The second part of the project is to classify the Abalones as "young" or "old". The machine learning methods applied for this binary classification problem are the following:

- Logistic Regression
- Support Vector Machines Regression
- Decision Tree
- K-nearest neighbors

We also used Cross-Validation in order to obtain some *tuning parameters*.

3 Data Visualization

First of all, we have plotted some histograms that represent the amount of data that belongs to a certain group. The command *hist()* does that data grouping is done by default. However, this only allows us to do histograms with float64 type data. The 'Sex' feature has object type data, so we have used *countplot()* command imported from *seaborn* library.

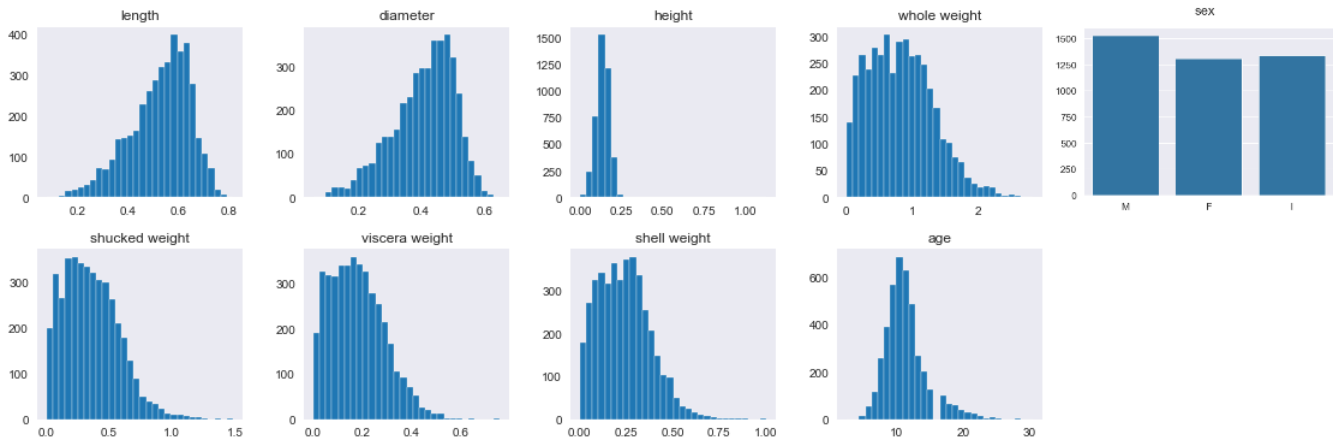


Figure 2: Histograms for all features

Moreover, we have studied some generative descriptive statistics of our dataset, such as the mean, the quartiles and the outliers. All that information is represented in the box plot of Figure 3. The red line in the middle of the box represents the median, which is actually the 2nd quartile. The box limits on top and in the bottom represent the 3rd and 1st quartiles, respectively. Moreover, above and below lines represent the maximum and minimum value of that feature, respectively, without taking into account the outliers. Finally, the circles represent the outlier values of each feature.

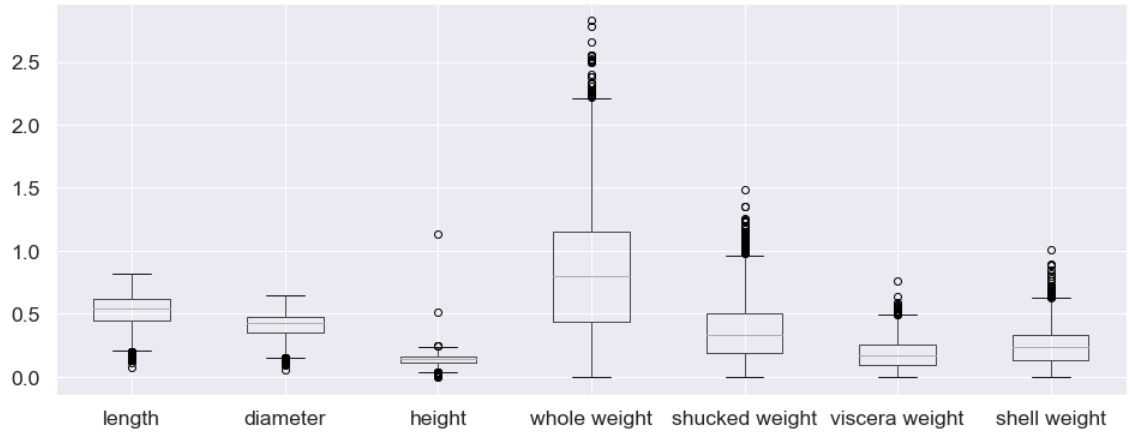


Figure 3: Boxplot for float type features

Then, we were interested in the correlation between the features. In particular, we have studied the correlation between the age and the other float64 type data features. In that way, we can determine whether the features are important when predicting the age. The correlation between two features take values from 0 (no correlation) to 1 (perfect correlation). From the results obtained (see Figure 4) we see that all correlation values are in the interval [0.42,0.63]. Therefore, we have used all the features in order to make the predictions.

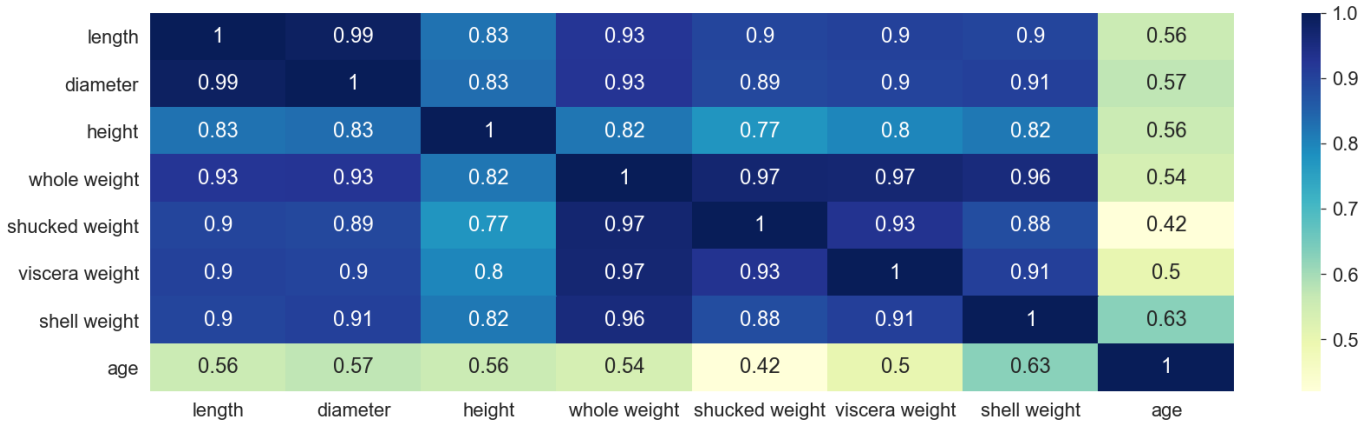


Figure 4: Correlation table

We have finally made some plots taking in all of them the age as one variable and the other features (one per plot) as the other. Therefore, we can check more visually the correlation that was presented in Figure 4. As an example, we present the plots concerning length, height and whole weight.

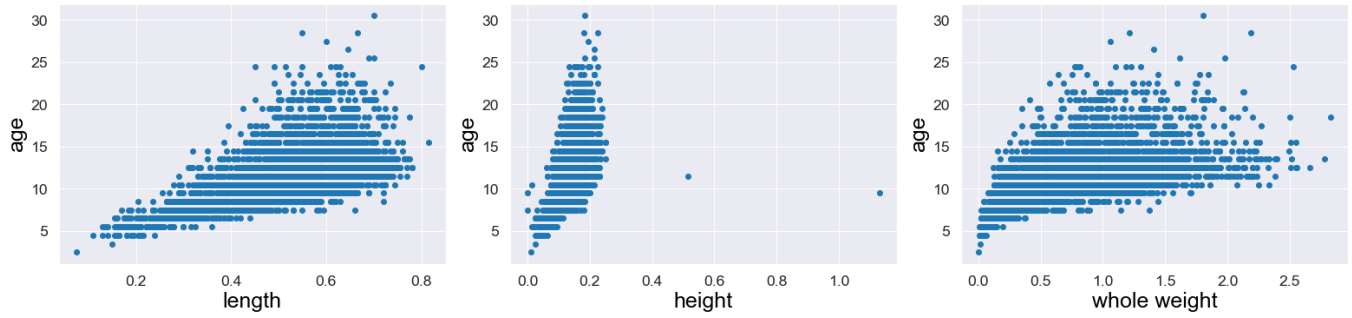


Figure 5: Correlation plots

4 Algorithms - Software

We now focus on the models we implemented in order to make the age predictions. We used a variety of supervised models to approach that problem. The purpose is to train the models in order to predict the age and test them, so that we can decide which one gives the best results. The age can be considered either as a discrete - by grouping our ages into subsets - or continuous variable. As a result, we can face the problem as both classification and regression.

4.1 Multiple Linear Regression

The multiple linear regression model is:

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_8 x_{i8} + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma),$$

where y_i is the dependent variable, x_{ij} are the independent variables, i is the number of observations, β_0 is the intercept, $\beta_1, \ldots, \beta_8$ are the coefficients for independent variables and ϵ_i are the model error terms with mean 0 and variance σ. The coefficients $\beta_0, \beta_1 \ldots, \beta_8$ are estimated in order to minimize the sum of squared residuals (SSR)

$$\sum_{i=1}^{8} (y_i - \hat{y}_i)^2,$$

where

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \cdots + \hat{\beta}_8 x_{i8},$$

being $\hat{\beta}_0, \hat{\beta}_1 \ldots, \hat{\beta}_8$ the estimated coefficients. Command: *linear_model.LinearRegression()*.

4.2 Ridge & Lasso Regression

Ridge and Lasso are regularization methods and their goal is to shrink the coefficients, so that accuracy of predictions will be improved and the variance will be decreased. This can be achieved by adding a *tuning parameter* λ, which is chosen with cross validation (Section 4.11). The idea is to estimate the coefficients to fit the data well, by making the SSR small and add a *shrinkage penalty*. That penalty is λ times the sum of squares of the coefficients, so that those who are too large are penalized. Therefore, the formula we want to minimize is

$$L_{\text{Ridge}} = \sum_{i=1}^{8} (y_i - \hat{y}_i)^2 + \lambda \sum_{i=1}^{8} \beta_i^2$$

In Lasso, the penalty is the sum of the absolute values of the coefficients. Lasso shrinks the coefficient estimates towards zero (similar to Ridge), but it has the effect of setting some variables exactly equal to zero, when λ is large enough. The tuning parameter λ is chosen by cross validation (Section 4.11) and, when it is small, the result is essentially the same as in the linear regression. As λ increases, shrinkage occurs so that variables that are at zero can be thrown away. Therefore, the formula we want to minimize is

$$L_{\text{Lasso}} = \sum_{i=1}^{8} (y_i - \hat{y}_i)^2 + \lambda \sum_{i=1}^{8} |\beta_i|$$

Commands: *Ridge()* and *Lasso()*.

4.3 Logistic Regression

Logistic regression models the probability that the label set belongs to a particular category, i.e. $\Pr(Y = k \mid X = x)$ called the *Bayes classifier*. In order the probability to give outputs between 0 and 1 for all values of X, we use the *logistic function*

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_8 X_8}}{1 + e^{\beta_0+\beta_1 X_1+\cdots+\beta_8 X_8}},$$

where $\beta_0, \beta_1, \ldots, \beta_8$ are the coefficients of the *multiple linear regression* model (Section 4.1). To fit the model, we use the *maximum likelihood* method to estimate the β's. That is, we choose the estimates $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_8$ to maximize the *likelihood function*

$$l(\beta) = \prod_{i:y_i=1} p(x_i) \prod_{j:y_i=j} (1 - p(x_j))$$

Command: *LogisticRegression(C=20).*

4.4 Linear Discriminant Analysis

We now generalize the logistic regression model to classify a response variable that has $K > 2$ classes. The new model is called *Linear Discriminant Analysis* and it is often used for multiple-class classification. We represent as π_k the *prior* probability and that a random observation belongs to the kth class and as $f_k(x) = Pr(X = x \mid Y = k)$ the *density function* of X for an observation that belongs to the kth class. Then, we can introduce the *Bayes' theorem*

$$Pr(Y = k \mid X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$

that gives us the *posterior* probability that an observation $X = x$ belongs to the kth class.

We assume that X follows a *multivariate Gaussian* distribution, with an specific mean for each class and a common covariance matrix. This distribution assumes that each individual predictor follows a normal distribution, with some correlation between each pair of predictors. We define the *multivariate Gaussian density* as

$$f(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right) \sim N(\mu, \Sigma),$$

being $E(X) = \mu$ the p mean vector of X and $Cov(X) = \Sigma$ the $p \times p$ covariance matrix of X. Therefore, the observations in the kth class follow a multivariate Gaussian distribution $N(\mu_k, \Sigma)$, where μ_k is the mean of X_k. The LDA model assigns an observation $X = x$ to the class for which the Bayes classifier

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$$

is the largest. Command: *LinearDiscriminantAnalysis().*

4.5 Support Vector Machine

The support vector machine (SVM) is an extension of the support vector classifier (SVC) that results from enlarging the feature space in a specific way using kernels. The SVC classifies a test observation depending on which side of a hyperplane it lies. The hyperplane is chosen to correctly separate most of the training observations into the two classes, but may misclassify a few observations. It is the solution to the optimization problem

$$\begin{aligned}
&\underset{\beta_0,\beta_1,\ldots,\beta_8,\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} M \\
&\text{subject to } \sum_{j=0}^{8} \beta_j^2 = 2, \\
&y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_8 x_{i8}) \geq (1 - \epsilon_i), \\
&\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C,
\end{aligned}$$

where M is the margin that we want to maximize, $\epsilon_1, \ldots, \epsilon_n$ are slack variables that allow individual observations to be on the wrong side of the margin or the hyperplane and C is a tuning parameter which bounds the sum of the ϵ_i's. The idea with the SVM is that instead of using the hyperplane, which is a linear boundary

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle,$$

we replace the inner product with a generalization of the inner product of the form

$$K(x_i, x_{i'})$$

which will be reffered to as a *kernel*. In our case, with the abalone data, we used the radial basis function (rbf) kernel:

$$K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^{P} (x_{ij} - x_{i'j})^2\right) \tag{2}$$

and by trial and error we found that the best gamma parameter is $\gamma = 0.5$. Command: *SVC(kernel='rbf', gamma=0.5)* for radial basis kernel.

4.6 SVMs with more than Two Classes

In its most simple type, SVM doesn't support multiclass classification natively. It supports binary classification and separating data points into two classes. For multiclass classification, the same principle is utilized after breaking down the multiclassification problem into multiple binary classification problems. Two of the methods based on this approach are the One-Versus-One and One-Versus-All classification.

4.6.1 One-Versus-One Classification

A one-versus-one approach constructs $\binom{K}{2}$ SVMs, each of which compares a pair of classes. In the abalone dataset, we have 28 classes. Therefore there will be $\binom{28}{2} = 378$ SVMs. We classify a test observation using each of the 378 classifiers, and we tally the number of times that the test observation is assigned to each of the 28 classes. The final classification is performed by assigning the test observation to the class to which it was most frequently assigned in these 756 pairwise classifications. Command: *OneVsOneClassifier(LinearSVC(random_state=0))*.

4.6.2 One-Versus-All Classification

In the one-versus-all approach We fit K SVMs, each time comparing one of the K classes to the remaining $K-1$ classes. In our case, there will be 28 SVMs and let $\beta_{0k}, \ldots, \beta_{8k}$ (since we have 8 features) denote the parameters that result from fitting an SVM comparing the kth class to the others. Let x^* denote a test observation. We assign the observation to the class for which $\beta_{0k} + \beta_{1k}x_1^* + \ldots + \beta_{8k}x_8^*$ is the largest, as this amounts to a high level of confidence that the test observation belongs to the kth class rather than to any of the other classes. Command: *OneVsRestClassifier(LinearSVC(random_state=0))*.

4.7 SVM Regressor

The SVM can be adapted for regression in ways that inherit some of the properties of the SVM classifier, thus getting the Support Vector Regression. We achieve this by introducing an ϵ-insensitive region (ϵ-tube) around the prediction function $f(x) = w^T x + b$. Then, SVR is formulated as an optimization problem by first defining a convex ϵ-insensitive loss function to be minimized and finding the flattest tube that contains most of the training instances, while minimizing the prediction error. This means that any point in our dataset that falls inside the tube is completely disregarded for error and we only take into account the points outside the tube.

Adopting a soft-margin approach similar to that employed in SVM, slack variables ξ, ξ^* can be added to guard against outliers. These variables determine how many points can be tolerated outside the tube and C is a regularization parameter that gives more weight to minimizing the flatness, or the error. This multi-objective optimization problem will have the form

$$\begin{aligned}
&\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \\
&y_i - w^T x_i \leq \epsilon + \xi_i^*, \quad i = 1, \ldots, N \\
&w^T x_i - y_i \leq \epsilon + \xi_i, \quad i = 1, \ldots, N \\
&\xi_i,\ \xi_i^* \geq 0, \quad i = 1, \ldots, N
\end{aligned}$$

Finally the solution, which is a hyperplane, is represented in terms of support vectors, which are training samples that lie outside the boundary of the tube.

The method described above is the Linear Support Vector Regression. Similarly to the SVM, instead of using the hyperplane, in order to gain higher accuracy we can also map the data to a higher dimensional space with the use of kernels, such as the radial basis function (2), for which the best gamma parameter is $\gamma = 0.043$. Commands: *SVR(kernel='rbf', gamma=0.043)* for radial basis kernel and *SVR(kernel='linear')* for linear kernel.

4.8 Regression Trees

Tree-based methods involve segmenting the predictor space into a number of simple regions and, in order to make the predictions, we use the mean of the training observations in the region to which it belongs. We take a partition of J regions of the predictor space and, for every observation that falls into one specific region, we predict its value by the mean of the values for the training observations in that region. In order to do that partition, we consider a *greedy* approach, known as *binary splitting*. We select the predictor X_j and the cutpoint s such that the two regions $\{X \mid X_j < s\}$ and $\{X \mid X_j \geq s\}$ lead to the greatest possible reduction in the *residual sum of squares* (RSS) given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

where $\hat{y}_{R_j}$ is the mean response for the training observations within region R_j. The process continues until a stopping criterion is reached. In order to avoid overfitting, we select a subtree that leads to the lowest test error rate. Command: *tree.DecisionTreeRegressor()*.

4.9 Classification Trees

In contrast to regression trees, we predict that each observation belongs to the *most commonly occurring class* of training observations in the region to which it belong. Moreover, we are also interested in the *class proportions* among the training observations that fall into that region. The way classification trees are done is similar to the task of growing of a regression tree, but instead of the RSS, we use the *entropy*

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk},$$

where $\hat{p}_{mk}$ is the proportion of training observations in the mth region that are from the kth class. This measure gets values near zero if the $\hat{p}_{mk}$'s are all near to zero or one. Command: *tree.DecisionTreeClassifier()*.

4.10 K-Nearest Neighbors

Given a positive integer K (number of neighbors) and a test observation x_0, the k-nearest neighbors (KNN) classifier first identifies the K points in the training dataset that are closest to x_0, represented by N_0. Then, it estimates the conditional probability

$$P(Y = j \mid X = x_0) = \frac{1}{K} \sum_{i \in N_0} I(y_i = j), \qquad I(y_i = j) = \begin{cases} 0 \text{ if } i \neq j \\ 1 \text{ if } i = j \end{cases} \tag{3}$$

Finally, KNN classifies the test observation x_0 to the class with the largest probability from (3). Commands: *KNeighborsRegressor(n_neighbors= 43)* for regression, *KNeighborsClassifier(n_neighbors= 22)* for multi-classification and *KNeighborsClassifier(n_neighbors= 27)* for binary classification

4.11 Cross Validation

Although it has nothing to do with prediction, we have used cross validation for the purpose of making our models more accurate. Some models, such as Lasso and Ridge Regression, have a tuning parameter that has to be set. Cross validation (CV) allows us to choose the best parameter in order to obtain the highest accuracy.

k-fold cross validation divides the set of observations into k groups, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining $k-1$ folds. The mean squared error, MSE_1, is then computed on the observations in the held-out fold. This procedure is repeated k times; each time, a different group of observations is treated as a validation set. The K-fold CV estimate is computed by averaging the values of test MSEs,

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} \text{MSE}_i$$

This procedure helps us to avoid overfitting. In our case, we separated the training set into 5 folds.

5 Results

Aiming to estimate the performance of the Machine Learning algorithms implemented, we used the train test split provided by the scikit-learn library. The test size we chose is 30%, meaning that it consists of 1254 datapoints. The results below are referring to the Abalone test dataset. Also, we added accuracy graphs which were made using plotly library to visualize and compare the performance of the different methods.

5.1 Regression

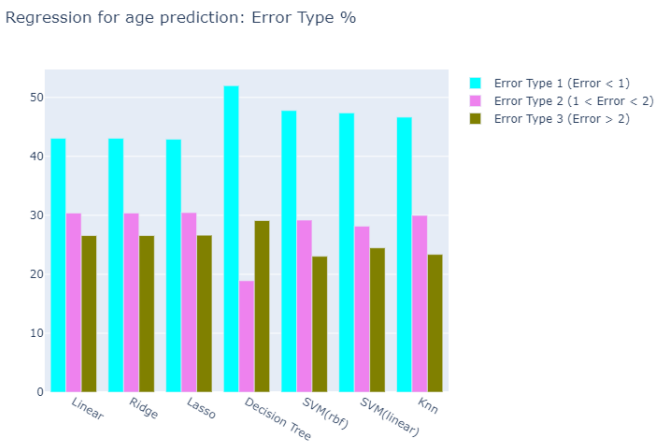
The first approach to the Abalone age prediction problem was with the use of Machine Learning regression algorithms. In order to quantify the accuracy of the models and assess their effectiveness, we categorized the errors in the following way:

- **Error Importance 1:** if error ≤ 1
- **Error Importance 2:** if $1 <$ error ≤ 2
- **Error Importance 3:** if error > 2

Moreover, we used the Mean Squared Error and the r2-score metrics (on the normalized data) for further evaluation of the different algorithms' performance. Keeping in mind the above, we will present the results of each regression algorithm:

- Linear Regression:
  - Error Importance 1 was reported in 540 cases (43.06%)
  - Error Importance 2 was reported in 381 cases (30.38%)
  - Error Importance 3 was reported in 333 cases (26.56%)
  - Mean Squared Error: 0.4682
  - r2-score: 0.5226
- Ridge Regression:
  - Error Importance 1 was reported in 538 cases (42.90%)
  - Error Importance 2 was reported in 382 cases (30.46%)
  - Error Importance 3 was reported in 334 cases (26.63%)
  - Mean Squared Error: 0.4677
  - r2-score: 0.5232
- Lasso Regression:
  - Error Importance 1 was reported in 538 cases (42.90%)
  - Error Importance 2 was reported in 382 cases (30.46%)
  - Error Importance 3 was reported in 334 cases (26.63%)
  - Mean Squared Error: 0.4681
  - r2-score: 0.5227

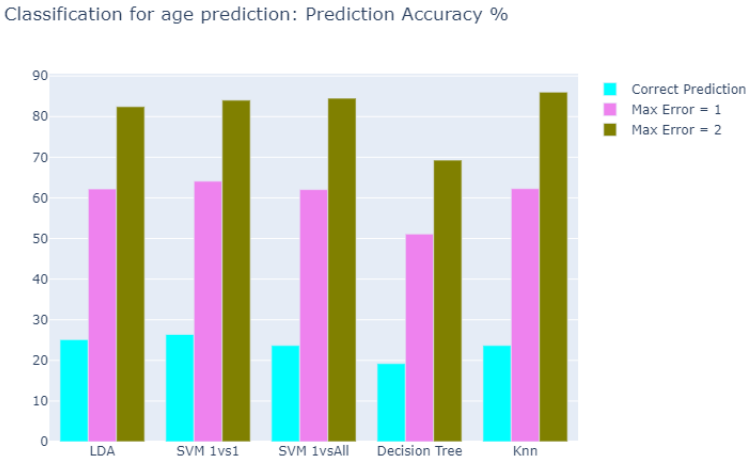
- Decision trees:
  - Error Importance 1 was reported in 630 cases (50.24%)
  - Error Importance 2 was reported in 253 cases (20.18%)
  - Error Importance 3 was reported in 371 cases (29.59%)
  - Mean Squared Error & r2-score are not suitable when working with the Decision Tree and therefore were not used for this regression method.
- Support Vector Machine with Radial Basis Function kernel:
  - Error Importance 1 was reported in 599 cases (47.77%)
  - Error Importance 2 was reported in 366 cases (29.19%)
  - Error Importance 3 was reported in 289 cases (23.05%)
  - Mean Squared Error: 0.4558
  - r2-score: 0.5353
- Support Vector Machine with Linear kernel:
  - Error Importance 1 was reported in 594 cases (47.37%)
  - Error Importance 2 was reported in 353 cases (28.15%)
  - Error Importance 3 was reported in 307 cases (24.48%)
  - Mean Squared Error: 0.4842
  - r2-score: 0.5063
- K-nearest neighbors:
  - Error Importance 1 was reported in 585 cases (46.65%)
  - Error Importance 2 was reported in 376 cases (29.98%)
  - Error Importance 3 was reported in 293 cases (23.37%)
  - Mean Squared Error: 0.4627
  - r2-score: 0.5282



5.2 Multi-class Classification

Another way to work with the Abalone age prediction is with the use of multi-class classification where the ages are divided in 29 classes ranging from 2.5 to 30.5. As we have already mentioned, the age is computed by adding 1.5 to the number of rings of the Abalone and therefore using the 29 ring classes ranging from 1-29 to make the prediction is equivalent to the original problem. In this project, we opted to predict the number of rings since then the data are integer numbers. For the purpose of evaluating the performance of the classification algorithms, we calculated the accuracy of the predictions but we also took into consideration the predictions that were off by 1 or by 2 since such small inaccuracy is acceptable in this case. The results from the different Machine Learning classification algorithms are the following:

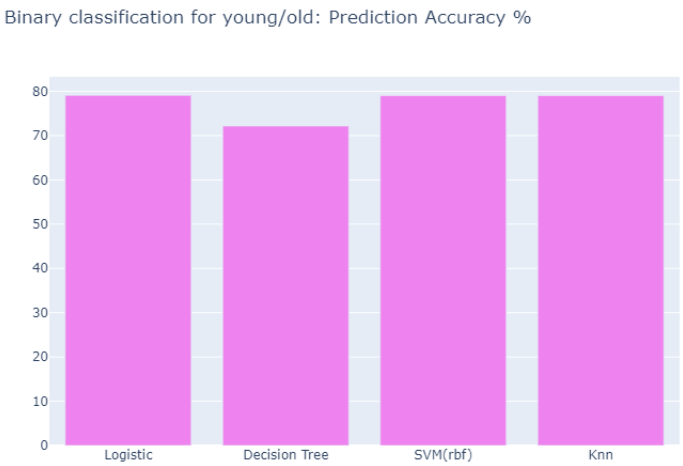
- Linear Discriminant Analysis:
  - Correct prediction: 25.12%
  - Maximum error 1: 62.20%
  - Maximum error 2: 82.46%
- Support Vector Machine One-Versus-One Classification:
  - Correct prediction: 26.40%
  - Maximum error 1: 64.11%
  - Maximum error 2: 83.89%
- Support Vector Machine One-Versus-All Classification:
  - Correct prediction: 23.60%
  - Maximum error 1: 62.04%
  - Maximum error 2: 84.53%
- Decision Tree:
  - Correct prediction: 18.02%
  - Maximum error 1: 50.40%
  - Maximum error 2: 68.58%
- K-nearest neighbors:
  - Correct prediction: 23.68%
  - Maximum error 1: 62.28%
  - Maximum error 2: 86.04%



5.3 Young/Old Binary Classification

The second part of the project was to implement Machine Learning classification algorithms to predict whether an Abalone is young or old. Same as before, we chose to work with the number of rings, considering an Abalone as young if it has less than 10 rings (meaning that it is less than 11.5 years old) and old otherwise. The results of each classification model are the following:

- Logistic Regression: correct prediction → 79.11%
- Decision Tree: correct prediction → 72.33%
- Support Vector Machine with Radial Basis Function kernel: correct prediction → 79.03%
- K-nearest neighbors: correct prediction → 79.03%



6 Conclusions

In this project, we worked on predicting the age of abalone snails using a Kaggle dataset of biological measurements acquired from the UC Irvine Machine Learning Repository. For the purpose of finding the optimal way to make the predictions, we approached the problem both as a regression and a classification one implementing different machine learning algorithms in each case. We also classified the abalones as young/old and implemented binary classification algorithms to predict in which category they fall into.

The regression analysis gave some very promising results, where some of the algorithms greatly outperformed others. Specifically, Decision Tree was very effective in accurate predictions since 50.24% of them were off by 1 year or less, while Support Vector Regression with Radial Basis Function kernel was superior in overall performance where 76.95% of the predictions had a maximum error of 2. Similar performance of that of the SVR was observed from the SVR with Linear kernel and the K-nearest neighbors.

In the second approach that involves multi-class classification, the algorithms, when having a small error margin of 2, were overall much more effective than the regression ones. Every model made the correct prediction less than 27% of the time but all of them (except for the Decision Tree) made an age prediction with maximum inaccuracy of 2 years 82%-86% of the time. In particular, given this error margin, K-nearest neighbors had an accuracy of 86.04%, outperforming the other algorithms, while Decision Trees offered very poor results with accuracy less than 70%.

Finally, the models used for the young/old classification problem had similar performance, except for Decision Trees once again which had around 7% lower accuracy than the rest. Specifically, Logistic Regression, Support Vector Machine with Radial Basis Function kernel and K-nearest neighbors all gave similar results at around 79% accuracy, where the Logistic Regression was slightly better than the rest with 79.11% accuracy. Decision Trees were worse at 72.33% accuracy.

7 Limitations

In general, all the predictions were quite accurate but we have to take under consideration the fact that some features which highly determine the lifespan of the abalone are missing. For example, certain characteristics concerning their ecosystem have a great effect in how long an abalone lives for. Another limitation that occurs, is that some features are highly correlated. For instance, the correlation between whole weight and length of abalone is 0.97. That fact can potentially raise the multicollinearity concern. For that reason we implemented the Ridge and Lasso regularization methods, where Lasso did not zero any of the coefficients.

Also, we have to point out that the infant label in the sex feature - the category of young abalones whose gender has not been determined yet - can possibly create a strong bias and harm the validity of our problem, especially affecting the young/old classification. Despite that, we did not remove any feature or observations, as we cannot assess the importance of each piece of data without an expert's help (for example biologist's). Therefore, we have to keep in mind that the accuracy acquired from the models may be notably weaker because of these limitations.

8 Bibliography

References

[1] Kaggle. https://www.kaggle.com/datasets/rodolfomendes/abalone-dataset

[2] G. James, D. Witten, T. Hastie and R. Tibshirani, *An Introduction to Statistical Learning*, New York, 2013.

[3] scikit-learn. https://scikit-learn.org/stable/

[4] Wikipedia. https://en.wikipedia.org/wiki/Abalone

[5] UBC MDS. https://ubc-mds.github.io/abalone_age_classification/Project_report_milestone2.html

[6] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning*, Stanford, California, August 2008

[7] M. Awad and R. hanna, Efficient Learning Machines, pp 67-80, Apress, Berkeley, CA, 2015

[8] Investopedia. https://www.investopedia.com/terms/

[9] ISLRv2. https://hastie.su.domains/ISLR2/ISLRv2_website.pdf

[10] Data camp. https://www.datacamp.com/tutorial/tutorial-ridge-lasso-elastic-net