Machine Learning - CISC 5800

Katie Wolf
Joseph Kass

Naïve Bayes Classification of Abalone as Compared to Decision Tree Classification of Abalone

Intro

The dataset we have chosen to use for this project is a series of weight, size and sex measurments of abalone. It was first published in December 1995 in a non-machine learning study of abalone biology by the Department of Primary Industry and Fisheries in Tasmania, Australia. Abalone are sea-dwelling snails that grow at an approximately linear pace of one inch per year for the first few years of their life after which growth slows significantly [1]. It is possible to predict the age of an abalone by counting the number of growth rings inside of the snail as they form ring patterns for each year that they are alive, similar to trees. The process for counting the rings is tedious, time consuming, and kills the abalone [2], so being able to predict the age without this intrusion could be beneficial. The dataset contains 4,177 abalone measurements, each consisting of 8 feature (Sex, Length, Diameter, Height, Whole Weight, Shucked Weight, Viscera Weight, Shell Weight, and Rings),

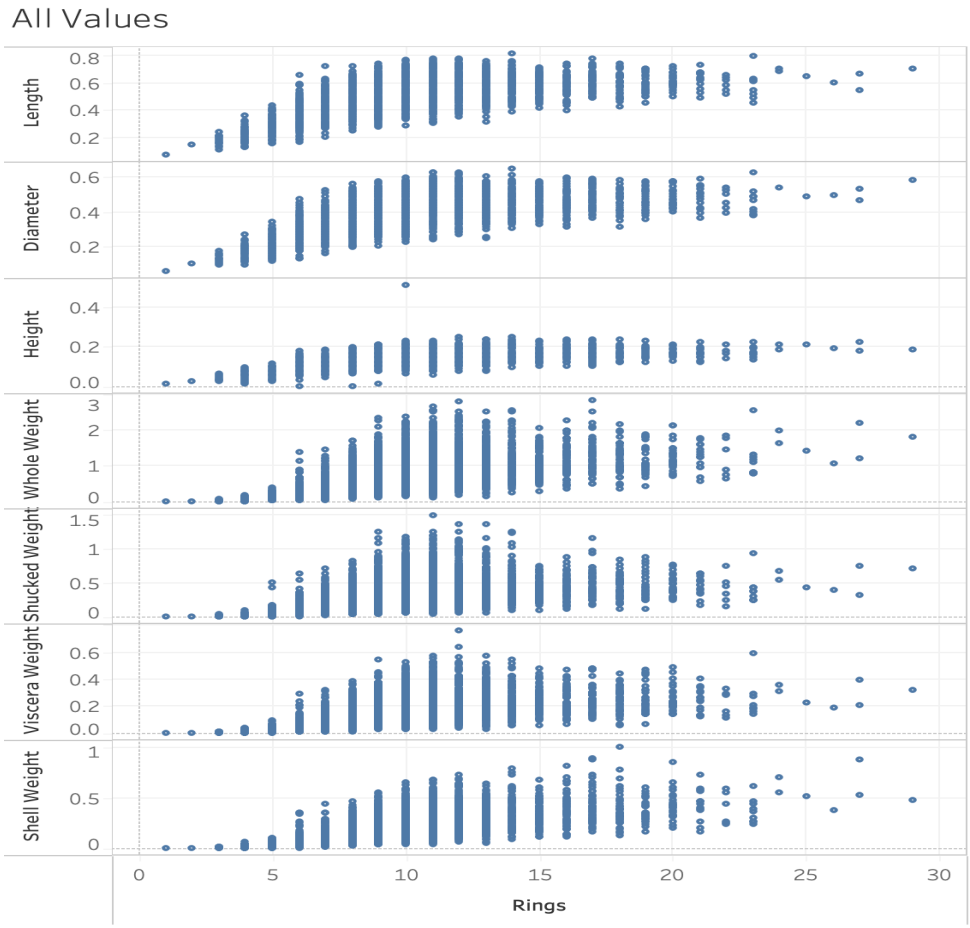
Our investigation of the data begins with a Naïve Bayes classifier, which we implemented with Gaussian, Uniform, and Rayleigh distributions. Naïve Bayes is not suitable for predicting a numerical response, so we have separated the age of the abalone into young, medium and old and will classify into these three buckets. The extension of the Naïve Bayes classifier will use feature selection and conversion of sex to integer values to investigate more accurate profiling. We have also implemented a Decision Tree using the Scikit-learn library and will investigate the effectiveness of training a Random Forest on the dataset.

Methods

Preparing the Data

The data required no cleaning before getting started. The data available in the UCI Machine Learning Repository has the missing values removed and the ranges of the continuous values have been scaled for use with an artificial neural network. This was done by the donor of the

database who divided the values by 200. We started by visualizing the data, to get a sense of what we were working with. Figure 1 shows how each feature relates to the number of rings an abalone has.



Rings vs. Length, Diameter, Height, Whole Weight, Shucked Weight, Viscera Weight and Shell Weight. The view is filtered on Exclusions (Height,Rings), which keeps 452 members.

Fig. 1 shows how the data is distributed across the different features being measured by this classifier. Generally all attributes grow as the abalone ages out of infancy, but after they reach close to 5 years of age, the association between an attribute and age becomes less stable. This feature of the data will influence how well our classifier does later in this report.

Despite being generally ready to work with from the outset, this data does not include an age attribute, but instead a "Ring" attribute that correlates closely with age. To better fit the data with the Naïve Bayes Classifier we created an attribute labeled “Category Age” and used that to

divide the data into age categories. We saw that age is distributed close to normally (see figure 2) and broke age categorization in three distinct groups: Young (<33%), Medium (33% < x < 66%), and Old (>66%), assigning the data points to the groups with labels 0, 1, and 2 respectively in the “Category age” column.

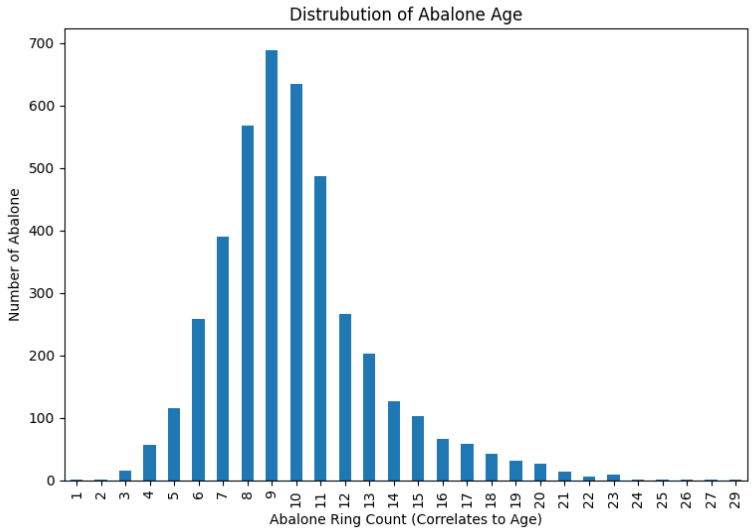


Fig. 2 This figure shows the distribution of ring counts (which correlate to age) in the dataset. It is a relatively normal distribution and because of that we felt safe splitting the abalone evenly into three categories to train the data.

Here are the two functions that work to split the dataset into categories:

```
def cat_helper(data, rings, bound_1, bound_2):
      if rings <= bound_1:
            return 0
      elif (bound_1 < rings <= bound_2):
            return 1
      elif (rings > bound_2):
            return 2

def cat_helper(data, rings, bound_1, bound_2):
      std = data['Rings'].std()
      Mean = data['Rings'].mean()
      bound_1 = norm.ppf(0.33, loc = mean, scale = std)
      bound_2 = norm.ppf(0.66, loc = mean, scale = std)

      data['Category Age'] = data['Rings'].apply(lambda x: cat_helper
      (data, x, bound_1, bound_2))
```

We ended up with the following bins for converting ring count to age:
Young: <= ~8.52, Medium: ~8.52 < x <= ~11.3, Old: > 11.3

While this type of categorical binning was necessary to adequately implement our classifier, it should be noted that the ultimate goal is to predict not just a general category of age but a far narrower band of potential ages to match the accuracy of age prediction provided by the current method (counting the rings). Having the results be more specific to age would be a good avenue to explore in future work.

Initial Implementation

After this age categorization, we built out our functions. We started with a separating function that takes the dataset and divides it into training data and testing data. This is done by randomly selecting a percentage of the data to be the training data. This percentage can be adjusted as needed and we chose a sample size of 10% for our implementation.

Next we had to obtain the mean and standard deviations necessary for our probability density function calculations. These measurements were required for every attribute we wished to use in our classifier. To start we used Length, Diameter, Height, Whole Weight, Shucked Weight, Viscera Weight, Shell Weight, and Rings. We excluded Sex as a classifiing attribute at the beginning because it contained non-numeric data, labeling each abalone as either M (male), F (female), or I (infant). Converting this singular feature into three features with numeric values is a method we eventually used to try to improve our classiifer, and that will be discussed later in the report.

Having decided on our initial classifying features, we calculated the mean and standard deviation for each of them. However, we couldn't just use the mean and standard deviation as calculated when using the entire training dataset as the input because then there would be no differentiation between the three age categories, preventing us from later classifying the test data. So, in addition to our summarizing function, we also have a separating function, which groups the dataset by the age category feature. Using those two functions as building blocks, we created a function that calculates the mean and standard deviation for each feature grouped by age category and returns those values in a dictionary. Now that we have all the values needed for the three continuous probability distributions we are testing in this classifier, we can start calculating probabilities.

The functions for each distribution all follow the same format - a function used to return the result of the probability density function associated with each distribution and a function that uses that calculation along with calculated prior probabilities to return the probability of a data point belonging to each of the three different age categories. This second function takes in a singular row of data, the mean and standard deviations previously calculated, and the entirety of the dataset. It uses the total number of data points and counts of how many data points belong to each age category to calculate prior probabilities. It then uses the numbers from the row of data and the associated mean and standard deviation for that value to calculate the probability of that row belonging to each category. It returns a diction with the probability for each category.

A final function chooses the age category label with the highest percentage and assigns that label as the guess for that row.

This is done for all three distributions: Gaussian, Uniform, and Rayleigh. The difference lies in the function where the probability density function is calculated, which is adjusted to match the distribution being used.

Once all the functions were built, we were able to test out our initial attempt at a classifier. We ran each distribution and measured our guess labels against the actual labels of the testing portion of our dataset. We recorded the percentage we got right and wrong for each, and wrote some small `if statements` to print out the distribution that had the highest percentage correct as the recommended distribution to use for further exploration.

We also added in a counter of how many data points were labeled correctly for each sex in order to see if that was an area where improvements could be made.

Extending the Classifier- Feature Selection

To further extend classification we have chosen to focus on the Gaussian distribution and remove Uniform and Rayleigh distributions, as the rings are approximately normally distributed and Gaussian was significantly more accurate in predictions.

The first method we implemented in extending the classifier was adding in feature selection. We wanted to see if certain features were better indicators or age than others, and if we could potentially exclude certain features to improve the accuracy of the classifier.

For the initial implementation of our feature selection, we again excluded Sex as a classifying feature, for the same reasons as before. We then looped through all the different combinations of our classifying features. This could potentially be an efficiency issue with other datasets, and the fact that we were only testing 7 features (resulting in 255 potential combinations) made this a feasible approach for us. This also allowed us to know which specific combination of features gave the most accurate results without having to worry that randomly removing and adding back in a feature at a time resulted in us missing a combination that may have been better. We also chose not to greedily remove features and keep them removed because of the small number of features we were working with. If we had chosen to remove two or three at a time, we would only be testing two new classification options, which we decided was not enough to get an accurate picture of potential for improvement.

With the feature selection method decided, we took the functions from our initial implementation and adjusted them to accommodate inputs with a varying number of features. Our original implementation was focused on our dataset and its size, and so the functions that calculated the mean and standard deviation for each row only worked with feature combinations that were similar in size to the original dataset. If a feature combination only included three or four features, those functions would break. They were built around excluding the 0 and 9th indexes of the data (the Sex feature and the added Category Age feature), which needed to be excluded when calculating the mean and standard deviation. We adjusted the functions to see what features were being tested and to remove the Sex and Category Age columns from the tested features if they were present, which worked for accommodating testing feature lists of all sizes and combinations.

As you'll see in our results section, this implementation wasn't very successful. However, using information from our initial implementation, we were able to see that using sex as a classifying feature had the potential to greatly improve our results.

Extending the Classifier - Converting Sex to a Boolean Value

To additionally extend the classifier we have decided to convert the value of Sex from M, F, I to three feature columns of boolean values which correspond to the three possible options for Sex. Pandas comes with a built-in method for converting categorical data into multiple boolean

columns. After applying pd.get_dummies, the Sex column becomes Sex_F, Sex_I, and Sex_M all with values of either 0 or 1.

By converting these categorical values, we are now able to use this data in combination with feature selection to observe that predicting based on sex greatly improves our gaussian predictor detailed in the results section.

Second Classifier - Decision Trees

For the second classifier of the abalone dataset we have chosen to implement a decision tree using the DecisionTreeRegressor classifier available in the Scikit-learn library. The data was prepared in the same method as specified above, including converting categorical Sex into three boolean columns.

To create the most basic decision tree we can provide only testing and training data with no specification for tree depth. If we do not provide a tree depth, the tree is automatically set to a depth of 26 which testing of a large range of depths proves to be the optimal depth. In order to test the accuracy of the Decision Trees we decided to review the results from a range of tree depths, from 5 to 105 in increments of 5 layers per run.

The second attempt of implementing a decision tree classifier made use of the RandomForestRegressor available from Scikit-learn, which takes multiple decision trees and averages the results of each to create the best final decision tree. This seems to be a better approach in general as the model is less likely to be overfit to specific peculiarities of the training set. The results from the RandomForestRegressor were slightly better than the basic DecisionTreeRegressor, shown in the section below.

Results

Our initial classifier had a run time of 9.82 seconds. The Gaussian distribution had an accuracy rate of 61.69%, the Uniform Distribution had an accuracy rate of 33.49%, and the Rayleigh distribution had an accuracy rate of 45.12%. The Gaussian performed significantly better, most likely because the age of the abalone in the dataset had a roughly normal distribution pattern (refer to figure 2 from earlier in the report to see this). While the success rate of the Gaussian distribution was better than the others we tried, it wasn’t the best and we hoped to improve.

Our feature selection tool allowed us to see the accuracy and run time of every feature combination, which allowed us to see if any combinations had better results. As could be expected, the smaller feature combinations tended to run much faster. For example, all feature combinations with two features being tested had a run time of less than 3 seconds. The trade-off for the quicker run time is generally a lower accuracy rate however. Some combinations did horribly - for example, height and weight as a feature combination had an accuracy rate of 20.64%. But, some smaller combinations, while not as good as our original implementation, did not do that much worse. Sex and Length, though, had a success rate of 59.40%, which isn't that far off from our initial implementation.

Ultimately, feature selection alone didn't end up resulting in an improved accuracy. The closest we got was a very similar accuracy rate, 61.7%, for a few different feature selection combinations. However, these combinations were smaller than our initial implementation feature list, resulting in a reduced run time. For example, the combination of Sex, Diameter, Height, Whole Weight, and Shell weight had the same accuracy as our original implementation, but a run time of 2.535 seconds.

While improving run time is good, we wanted to improve accuracy as well. Using our original implementation, we were able to see the potential benefits of using Sex as a classifying feature. All three distributions we tested had interesting accuracy results when looking at their ability to label a data point correctly when the accuracy was broken down by sex. See the following table for accuracies:

Distribution/Accuracy	% Correct - Female	% Correct - Male	% Correct - Infant
Gaussian	52.46%	55.36%	77.9%
Uniform	14.24%	19.62%	68.06%
Rayleigh	39.57%	39.82%	56.54%

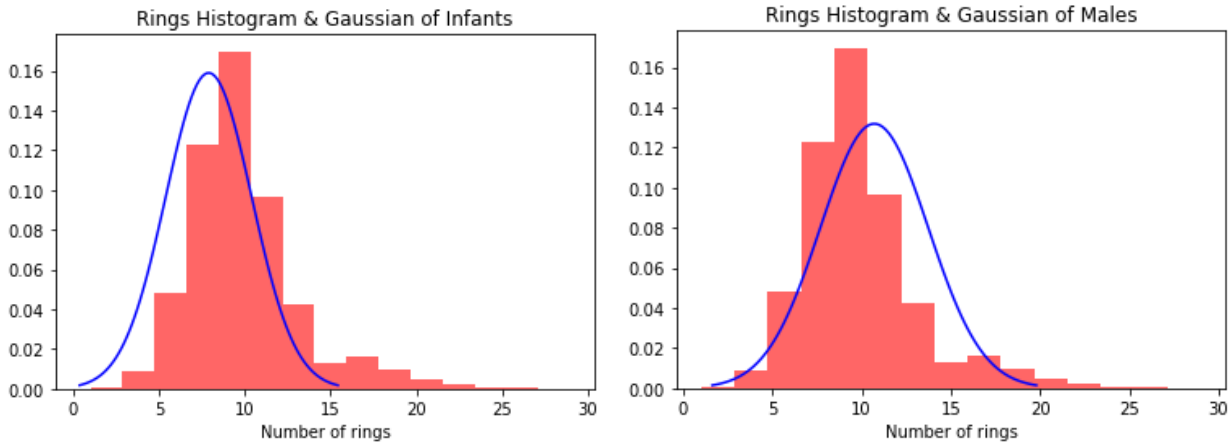
Figure 3. This table displays the accuracies each distribution had when categorized by sex. This is when not using Sex as a classifying feature. Instead, it is the initial results divided up into categories based on Sex.

As demonstrated by Fig. 3, every distribution worked the worst when classifying female abalone. They did a bit better when classifying male abalone. All three did very well when classifying infant abalone. If you refer back to Fig. 1, you can see that abalone tend to grow rapidly while they're

still under the age of about 5 to 8, and after that it becomes much harder to see discernable growth. It also happens that abalone aren't able to have their sex determine until around that age as well, meaning that Infant is an option for Sex categorization. So you see a strong ability to classify for that "Sex" category, because it is the most unique and the most closely related to the actual age of the abalone.

Because of these aspect of the dataset and of how abalone Sex categorization works, we wanted to try to add in Sex as a classifying feature.

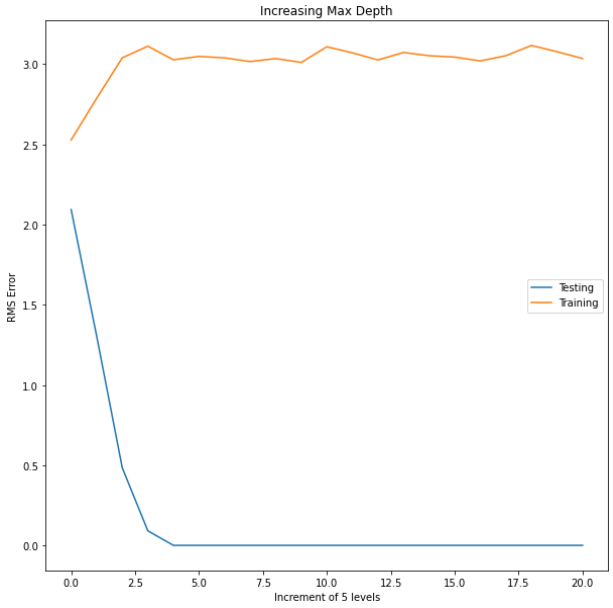
When combining feature removal with conversion of categorical sex to a boolean value, we are able to determine a sequence of features that creates 100% accuracy in prediction. One of the winning combinations of features is Height, Whole weight, Shucked weight, Viscera weight, Shell weight, and Sex_I which excludes Sex_F, Sex_M, Length, Diameter. This combination also appears to require the least number of features, but other combinations that use either only Sex_I or Sex_M can reach 100% accuracy in prediction. When we consider that Sex_I denotes an abalone that is too young to determine the sex, it is obvious that if we include this feature we would have a very accurate predictor for the Young categorization. It is less obvious why Sex_M would have a similar effect, but by removing the possibility of the Young categorization the model is effectively mirroring the predictor created when selecting for Sex_I. The below graphs demonstrate how selecting for Sex_I or Sex_M will create more accurate predictions by giving the predictor a starting estimation that favors either Young, or Medium and Old.

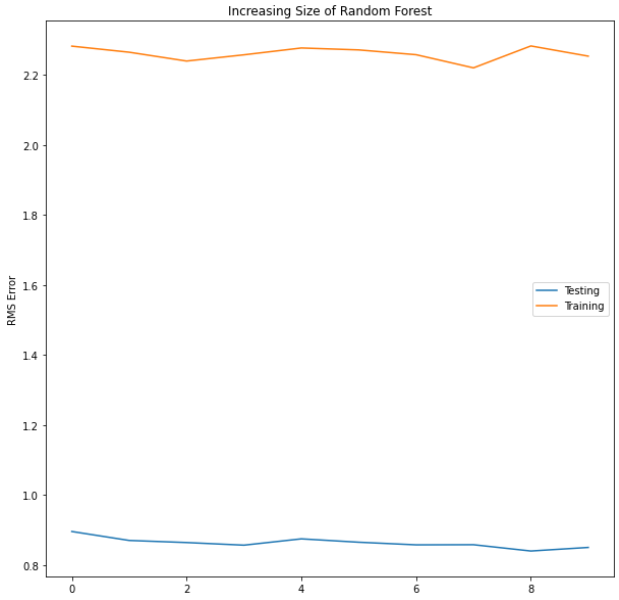


These predictors took 13.48s to run.

The results from the implementation of the decision tree were somewhat accurate when using the random forest regressor, while iterating the depth of a basic decision tree seemed to have limited or negative effects on the classification of abalone age. As a decision tree is easily understandable through reading each level of the tree, it is possible to confirm what causes the issue for each misclassification in a simple system but is logistically impossible to scan entirely through multiple trees with significant depth. To address this issue, we have taken the measure of Root Mean Squared Error to confirm which trees are performing the best.

The below graphs show the effects of increasing the depth of the decision tree have a negative effect on classification which is likely due to overfitting on the training data. Given that the library chooses a depth of 26 automatically, increasing the depth to 100 is likely too finely tuned to accurately predict the age. If the data set consisted of more features and had more training data it is possible to implement a deeper tree, but increasing the depth by an order of magnitude does not benefit the model. The Random Forest has better accuracy in general, but also does not appear to benefit from increasing the size of the forest. Overall the size of the error is small, so both classifiers show value in developing further.





These predictors took 5.49s to run. Depending on the size of the dataset this may be a better option for computational constraints

Conclusion

While the decision tree method seems the fastest and least computationally taxing, it isn't quite as accurate as the results offered by the bayesian classifier with sex as a classifying feature. This was generally the theme of our testing overall - the quicker the code or the more simple the implementation, the less accurate the results ended up being. Keeping sex out of the classifying features made the code simpler and required less manipulation of the data, but also prevented the results from being as accurate as they could have been. We had the same observation with converting age from discrete numbers to three category bins - this made classification easier, especially for our chosen classifiers, but our results only tell you whether an abalone is young, middle aged, or old, which is not as accurate as providing a numerical answer.

It's apparent that when deciding how to structure and build a classifier, there are some internal considerations required when assessing the dataset - how big is the dataset, how complex is it, is accuracy the most important thing or can some accuracy be sacrificed for the sake of improved run times and computational power required to run the script.

---

[1] California Department of Fish and Wildlife. "Saltwater Fish Report for 10-13-2011." Eastern Sierra Fish Reports. https://www.easternsierrafishreports.com/fish_reports/41015/how-fast-do-abalone-grow?.php#:~:text=Abalone%20grow%20nearly%20one%20inch,years%20to%20grow%20another%20inch (accessed May 4, 2022)

[2] S. Waugh. "Abalone Data Set." UCI Machine Learning Repository . https://archive.ics.uci.edu/ml/datasets/abalone (accessed May 4, 2022)