Alexander Bae
DATA 71200
Professor Devaney
2021 June 30

Machine Learning and Abalones

In a packed, yet illuminative four-week course, we applied the principles of machine learning onto data sets of our choosing. The basis for me choosing the dataset that I did was attributed to mostly circumstances at the time; I had been suffering stomach pains throughout the day and prepared myself a bowl of abalone porridge to settle my stomach. While I was eating the porridge, I stumbled upon a dataset about abalone on the University of California Irvine's machine learning repository and decided immediately that it would be the basis of my project.

The dataset was relatively straightforward in terms of content. It contained eight different attributes, with most of the attributes pertaining to weight or physical size. Only two of the attributes had nothing to do with those characteristics: the number of rings (which would help to determine age), and the sex of the abalone itself. The physical size and weight attributes were of the float data type, while the number of rings, and the sex were integers and strings, respectively.

In addition to being straightforward, the data was extremely clean, and contained no missing data (this was already handled by UCI). There was little to do terms of data cleaning and wrangling/munging, since the task of imputation was unnecessary. Instead, the following tasks were required:

- Converting the number of rings to abalone age (as directed by the UCI abalone dataset page)
    - This was achieved by creating a new age column by using the syntax abalone_df['age'] = abalone_df['rings'] + 1.5. The age column will also be a float as opposed to an integer.
    - The age column meant that the rings column was no longer necessary, so the entire column was discarded using abalone_df.drop('rings', axis = 1)
- Performing one-hot encoding and transforming the character strings for sex into separate columns encoded by 0s and 1s depending on the abalone's sex.
    - Infants were filtered out for the supervised learning models because they skewed the data, and learning models such as regression are heavily affected by the presence of outliers.

While the UCI website stated that the dataset was used to help predict the age of the abalone based off its measurements, I wanted to approach it from a different angle because of the large number of projects out there that utilized this same dataset to predict for the same target feature. Instead of age, I decided to see if the dataset could be used to predict the size of an abalone given other physical attributes for the supervised learning models, and whether the dataset could help unsupervised learning models accurately predict what sex the abalone would be given its physical attributes.

Visualizing the distribution of the data yielded some interesting results. The histogram for length displayed a left skew, which made sense because nearly one-third of the observations belonged to abalone classified as infants as opposed to male or female. However, all the weight

distributions showed the opposite; there was a right skew for all of them, which implied that most of the abalone, even adults, did not reach their full size.

The scatter plots featured a variety of curves, depending on the attributes that were plotted along the x and y axes. For example, shell weight had a positive linear relationship with total weight, which made sense, as one was a subset of the other. Total weight had a logarithmic plot when graphed along with shell length (diameter). The patterns remained relatively constant even with the various transformations applied to it, like squaring, cubing; etc. Height showed either an extremely horizontal curve with little growth, or a very vertical line with little length; this was because abalone are known to be very flat creatures, with extremely low height, even as adults. The boxplot displaying ranges confirms this, with height being the feature with the lowest range of values, and outliers.

With the preprocessing and data exploration steps finished, the machine learning algorithms could be applied to the data. The algorithms chosen for the supervised learning models were regression, and random forest. Each of the algorithms will be discussed in further detail in the sections below.

Regression was chosen because it was the most fundamental and straightforward learning model available. I selected it primarily because I knew that my target variable was a continuous one, and regression was one of the models best suited for that job. For every variable(s) x, there would be a y, using the formula $y = a + b(x)$, with b being the slope of the line, and a being y when x is equal to 0. There was some extra pre-processing to do, as the features all had different ranges, as displayed on the data visualizations. They needed to be scaled accordingly in order to better address the different ranges, and different outliers in each of the attributes. Once the feature scaling was completed, it was a matter of splitting the data into training and test sets (at a

standard 80:20 ratio) and fitting (and predicting) the data. Finally, I assessed the mean absolute error, and the root mean squared error between the predicted values of the fitted model, and the values present in the test set, to determine how strong the model would be at predicting the target variable. The R^2 value was approximately 0.959, which meant that about 96% of the variance in the data's values could be explained thanks to the variables, which was a strong early indicator of the model itself, but further testing needed to be done to ensure that the model wasn't overfitting and memorizing the training set instead.

Because of this concern about overfitting, I decided to implement cross-validation to ensure that my results would be consistent throughout regardless of the number of partitions used. I decided to do this twice, separating the data into 5 parts, and 10 parts. Performing the cross validation and examining the results showed similar consistency to the results performing without cross validation, so it was a good sign that the model was working without overfitting.

The random forest regression model was different compared to the linear regression models. For one, we did not need to perform any feature scaling on the data, since it is tree based, and outliers wouldn't have as much of an effect on the model. I didn't specify parameters when instantiating a new RandomForestRegressor object, if only because I wanted to run it with default parameters first before performing hyperparameter tuning. Like the linear model before it, the random forest regression model performed well on the training and test data, but this model is particularly susceptible to overfitting, so I needed to perform cross validation once more, along with hyperparameter tuning.

It was here that I became all too aware of the importance of performance issues, and runtime when executing code, as when I attempted to perform a random grid search, the process took almost an hour to complete with parameters that were set too high. As such, I had to lower

the parameters set to a more manageable level, but I was concerned that I was too invested in saving resources and runtime and giving up on output quality. My concerns were valid too, because the model's accuracy improved very slightly at the end of the tests, meaning that the using default parameters yielded results that were comparable to the results using the best possible parameters.

The process for PCA and unsupervised learning took a different turn, as I wanted to explore whether the gender of the abalone could be predicted. Due to my limited understanding of the algorithms at the time, I did not think that I could use these to predict the same target variable as the supervised learning models did, and for that reason I changed the target from shell length to abalone sex (from a continuous variable to a categorical variable).

With PCA, I wanted to identify the components that contributed the most heavily to the output's variations. The initial PCA test with the number of components set to 2 accounted for about 92.7 percent of the variation, so I decided to add a third component. With the third component, 95.9% of the variance was explained, so that was the level I decided to keep moving forward. However, initial visualization of the data based off class revealed that there was significant overlap and minimal distance between the groupings. Even though I didn't have the tests conducted, I surmised that algorithms such as k-nearest neighbors were not going to work well.

Fitting the KNN model to the data proved my concerns to be true, as the model operated under the assumption that every data point that was near another data point (based off Euclidean distance) was of the same class. Because the first visualization showed that there were many classes near one another, the model had a difficult time learning. This was shown by the ARI and the silhouette scores, which were extremely low, in both the default tests, and tests with the

data modified by PCA. However, it was interesting to note that it was able to distinguish infant abalone better than male or female abalone, because infant abalone had their own distinct portion of the data distribution whereas the other two classes were spread throughout the entire scatterplot.

However, agglomerative/hierarchal clustering proved to work better, although its scoring was still a far cry from being a reliable model. I believe it worked better than k-nearest neighbors because it worked as a bottom-up approach. Each data point started as its own cluster but was combined rapidly with other data points/clusters until the end results are the top-level groups (in this case, male, female, infant). It is similar to how phylogeny trees in biology and species classification tasks are performed; a species can be combined with other similar species to form into a genus, into a family, and so on, until the top level (kingdom or domain) is reached. Working with PCA data improved the scores of the model by nearly 33 percent, as well, going from about .45 to about .63 overall.

Finally, for DBSCAN, I wasn't sure how well it would do compared to the other two. It operated by comparing the areas that were most densely populated to the areas that were the least. With that in mind, I guessed that it would learn about the infant group the best, similar to the agglomerative model. However, the problem was that because of the reliance on distance from one data point to another, the densest part of the data distributions were the ones that had abalone of all three groups in proximity to one another. The model had difficulty establishing different clusters because it keeps counting points until it comes across a data point that lies outside of its established range, which is when it creates a new cluster. That is why the resulting graph only showed four clusters. The result scores also show that this was not a model suited for the data as well.

In retrospect, attempting to do things on my own instead of following established directions proved to be a messy and disastrous affair for the unsupervised learning methods. The supervised learning methods proved to be much more straightforward and yielded better results. There are various aspects of the project that could be changed if I were to attempt it again. For one, I'd probably change the classes of the abalone sex category, and group both male and female abalone together as "adults" and see if the models could differentiate between adult and infant abalone better. Another would be to attempt the work on a much more powerful machine, as the computational power of the machine I was used was insufficient and at times, threatened to freeze my machine completely. Finally, knowing what I do now, I would try and see if the unsupervised learning models worked well on regression models, since I found out after submitting the project that unsupervised learning models can work for regression type problems where the target column is a continuous data type.