Predicting Penguin Species with Machine Learning

Emma Kate Garrett, Savannah Martin, and Evan Fouss
CIS377 - Group Project
evan.fouss@students.fhu.edu, savannah.martin@students.fhu.edu, emmakate.garrett@students.fhu.edu

1 Introduction

Given the previously gathered penguin data metrics, our group set out to train and optimize a machine learning model to predict a penguin species. With statistics such as culmen length and depth, flipper length, and mass, we are able to fit data to a model and teach an algorithm.

Today, various techniques can be implemented to solve these types of problems within artificial intelligence. Various algorithms such as decision trees, support vector machines, and neural networks are utilized in order to achieve an optimal solution. These practices are limited to the resources available, data that has been collected, and the tools created.

If we are successful, we can positively impact penguin research and make a difference by providing wildlife scientists a tool to better understand these species. Using previously gathered online data, we were able to manipulate vital information about these penguins. The most important aspects of our data included the culmen lengths and depths, the flipper lengths, the mass, the island of origin, and the stable isotope values of carbon (delta13C) and nitrogen (delta15N) in the penguins. While the data did record a few NaN values, we were still able to train the model appropriately.

2 Approach

To solve this problem, we attempted training three different machine learning models. Our models of choice included K Nearest Neighbors, Decision Trees, and a Support Vector Machine. By cleaning, normalizing, and training the data, we were able to implement these three different solutions to this problem.

From our knowledge, these machine learning algorithms have been extremely useful to fit data and predict outcomes. These approaches could be considered standard and practical ways to solve this problem.

When thinking about problems we might anticipate, it was important to remember the size of our dataset, the naming conventions, and the gaps in our data. This might lead to models that could be prone to overfitting as well as other potential roadblocks.

Some problems we encountered during the processing of the dataset. Upon reading the csv file, we were required to drop unnecessary columns as well as rows containing illegitimate values. The first attempt at cleaning this dataset did not work for us, and it required multiple attempts to appropriately handle our dataset.

3 Experiments and Results

Once the dataset was chosen, as a group we decided to do three different models. Once we normalized the data, the process to iterate over different models was easy. The three models we chose were decision trees, SVMs, and KNNs.

The reason we chose a decision tree was because it seemed like it would be fun to implement, but also decision trees can be very finicky to deal with. They can be hard to optimize due to the limitations of the model. This seemed like a good choice because it would help us to better understand what type of data we had.

We chose an SVM because it was completely different from the decision tree. We wanted to be able to show what a good model was and the comparison from it to a bad model. It was easy to implement as well, and easy to tell the differences from it to the other models.

Lastly we decided to go with a KNN model. The other choice was a neural network, but we decided that might be a bit involved, and overall it turned out to be the best choice. This was because it somewhat acted like a control group in our experiment. Seeing that it was used in the Kaggle paper, it worked as a solid control for our other models to base off of. We did further digging in to the KNN to see why it did so well, and it turned out to be a good eye opener to the dataset we had.

Below are the links to the sources used for the models below.

penguin dataset : The new Iris | Kaggle

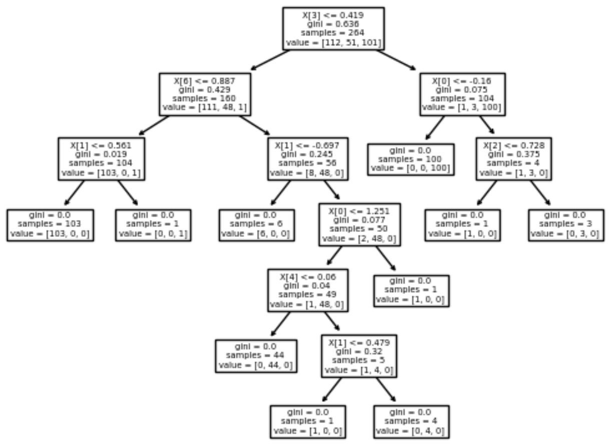
https://github.com/savannah-martin/fightingpenguins

3.1 Decision Tree

From study, it has been tested that a decision tree is a very useful and prevalent model in machine learning to predict and classify outcomes. These models are typically fast and easy to use on large data sets.Using a Decision Tree (Figure 1), we measured success based on three metrics: accuracy, precision, and recall. The results of this method yielded high percentages in all three categories. The quantitative data consisted of values over ninety-five percent (0.95) in all of the mentioned areas. This is noted in the code in the repository linked as well as visualized in Figure 2.

Conceptually, our results proved to be very accurately trained on our decision tree model. This use of this machine learning algorithm could be considered a success.

However, it could be noted that there was a relatively limited amount of data used to train this model. This could cause the model to veer towards overfitting which is often a side effect of small datasets.



X[3] <= 0.419 gini = 0.636 samples = 264 value = [112, 51, 101]
X[6] <= 0.887 gini = 0.429 samples = 160 value = [111, 48, 1]
X[0] <= -0.16 gini = 0.075 samples = 104 value = [1, 3, 100]
X[1] <= 0.561 gini = 0.019 samples = 104 value = [103, 0, 1]
X[1] <= -0.697 gini = 0.245 samples = 56 value = [8, 48, 0]
gini = 0.0 samples = 100 value = [0, 0, 100]
X[2] <= 0.728 gini = 0.375 samples = 4 value = [1, 3, 0]
gini = 0.0 samples = 103 value = [103, 0, 0]
gini = 0.0 samples = 1 value = [0, 0, 1]
gini = 0.0 samples = 6 value = [6, 0, 0]
X[0] <= 1.251 gini = 0.077 samples = 50 value = [2, 48, 0]
gini = 0.0 samples = 1 value = [1, 0, 0]
gini = 0.0 samples = 3 value = [0, 3, 0]
X[4] <= 0.06 gini = 0.04 samples = 49 value = [1, 48, 0]
gini = 0.0 samples = 1 value = [1, 0, 0]
gini = 0.0 samples = 44 value = [0, 44, 0]
X[1] <= 0.479 gini = 0.32 samples = 5 value = [1, 4, 0]
gini = 0.0 samples = 1 value = [1, 0, 0]
gini = 0.0 samples = 4 value = [0, 4, 0]

Figure 1 - Decision Tree

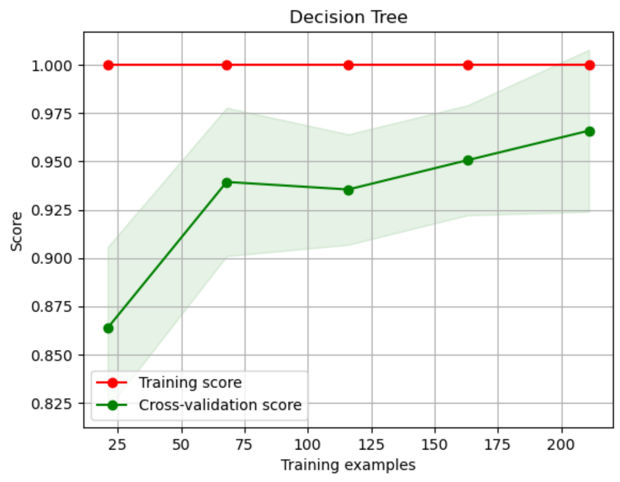


Figure 2 - Decision Tree Plot

3.2 SVM

Using the support vector machine (SVM) method, we measured success using accuracy, precision, and recall. The SVM method proved to be the least successful out of the three models. All three measures of success were below .5 when using the macro, micro, and weighted average.

I would consider the SVM a failure for this specific dataset because the results were so low. This method was not ideal because there were so many gaps in the data, and SVM's have trouble compensating for the lack of information. As shown in Figure 3, in order for the SVM score to be above .95, it required more than 125 training examples, where in other models, the score rose higher, quicker with fewer training examples

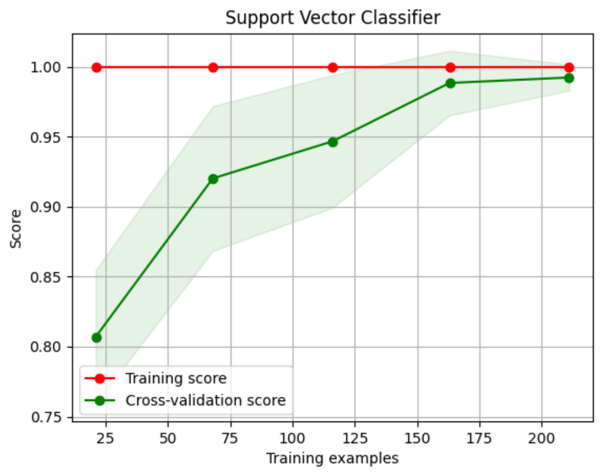


Figure 3 - SVM Plot

3.3 KNN

Below are the experiments involving the K nearest neighbors, or KNN, plots. KNN was chosen as a model because it made sense for the type of dataset used. The Kaggle paper and repos referenced gave a clue to use this type of model, and so using code from past student experiments, KNN plots were created.

One of the things to keep in mind is that the average of the plots had to be set due to the fact that multiple different types of metrics are being solved for.

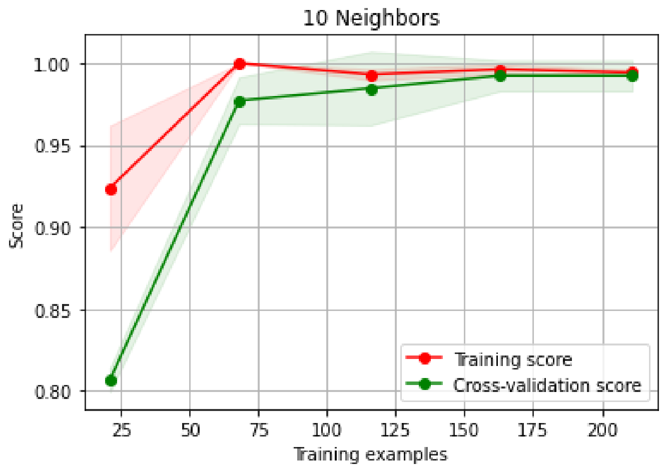


Figure 4 - KNN Micro Plot

The first type that the average was set to was micro. This totals the true positives in the data and divides by the total positives. The recall, precision, and accuracy all are equal to one, so essentially the model perfectly represents the data metric wise

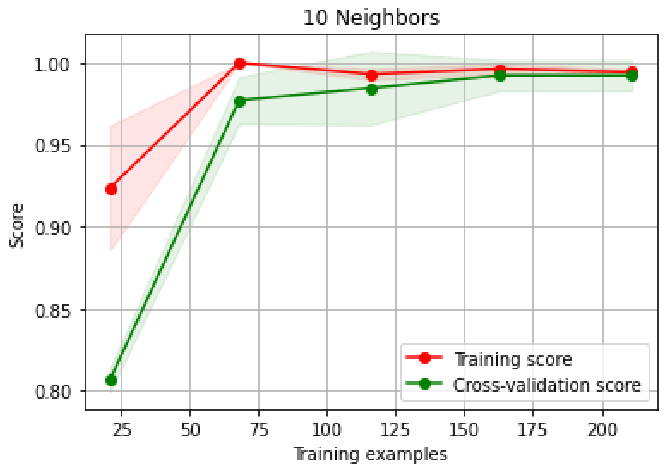


Figure 5 - KNN Macro Plot

The second type the average was set to was macro. This takes the average of the recall classes. The metrics for this were also exactly one in every category.

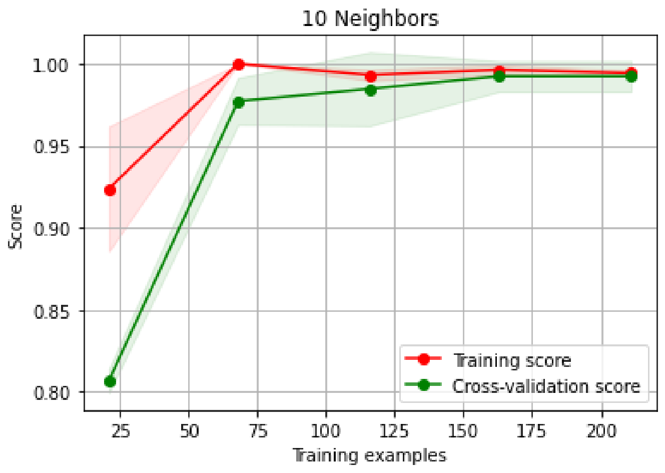


Figure 6 - KNN Weighted Plot

Last but not least the average was set to weighted. This takes the sum of the individual array elements and divides by the total sum of the weights. To note, the metrics are again one.

As one can see, the graphs are very similar to each other. This means that the type that the average is set to does not really matter, it is up to the user and what type of measurement is desired to be taken.

KNN is very simple, and allows a wide range of measurements to be taken in a very simple manner. The code I used ran loops to do multiple tests to see if any difference happened as the number of neighbors increased. Again, the more neighbors used the harder it can be to do KNN measurements. KNN is simple to set up and simple to iterate over, and easy to optimize if the data fits a KNN model.

The Kaggle paper did have a KNN run over it, and it makes sense. The results show that a KNN model is great, given they all show a perfect one-to-one representation of the data. This may be slightly off-putting since models usually aren't this nice to work with. However, since the data set is small there is little reason to suspect over-fitting from a glance.

4 Additional

The data that was used required normalization and cleaning. We wanted to write a model that could optimally predict the species of penguin based on the information given.

After observing all of the models and their results, the KNN proved to be the most ideal model. The decision tree was the second best model, and the SVM was the least beneficial.

This problem was structured in a simplistic manner. Given previously gathered and organized data, we were able to use machine learning techniques to fit this data and make our predictions. Our models were chosen based on the advantages they provided. We were able to work with the structure of our dataset most effectively using the algorithms best suited to handle this problem.

Using a pandas library, we were able to manipulate the dataset and read in the data. From there, we normalized the data, encoded the string values, dropped unnecessary values, and removed NaN entries. For the KNN model, it yielded 100%, and the Decision Tree produced high metrics as well. This could be seen as overfit, and it could tend to not generalize well.

For the Decision Tree, the random state hyperparameter was used to optimize the model, for the KNN, the n neighbors, and for the SVM, the kernel rbf was used. They were chosen through trial and error, picking the optimal number for our metics. Overall, the quantitative performance varied little. These models were created based off the existing code and models we created throughout the course of our class. These starting points helped us to have a basis for how to approach and solve our problem.

5 References

Listed below are the relevant works that inspired our project. It includes articles used to make decisions about models and hyperparameters as well as some insight into methods and approaches to manipulate the data.Posted on Kaggle, we found the penguin dataset we used to implement our models.

analytixlabs.co.in/blog/decision-tree-algorithm/
https://www.geeksforgeeks.org/decision-tree/
penguin dataset : The new Iris | Kaggle
Penguin Dataset (KNN) | Kaggle
Penguin dataset - EDA and classification using KNN | Kaggle
Palmer station penguins classification using machine learning