Palmer Penguins

Louis Bailey, Morgan Buterbaugh, Lexi Neese

MATH 447/547

December 2022

**Introduction**

An increasingly popular way to analyze and classify data on a large scale is to use machine learning classification. This artificial intelligence method is continuously growing, due to its ability of allowing expansion within data operations. A classifier is defined as an algorithm that assigns data points to distinct classes. There are two subgroups of classifiers, one being supervised, the subgroup we will focus on, and the other being unsupervised. Supervised classifiers train a data set to distinguish labeled and unlabeled data in order to recognize patterns and classify unlearned data [7].

Machine learning operators can be used to model a data set and simulate different test-train situations. There are many different machine learning software’s, however we will focus on Python, R Studio, and MATLAB. Within these software’s we will look at KNN, or k-nearest neighbors, multiple linear regression, and decision tree classifications, respectively.

In order to train, a data set is needed. For this study, we will be using the Palmer Penguins data set, which is a popular data set used to get more comfortable with machine learning. We will test-train the data and simulate how accurate our classification models are for each machine learning software to determine which software and classification model provides the best results.

**Data**

For this study, we are using the Palmer Penguins data set. The Palmer Penguins data set was collected from 2007-2009 by Dr. Gorman and the Palmer Station Long Term Ecological Research Program in Palmer Archipelago, Antarctica. It contains information including flipper length, bill length, bill depth, body mass, island found on, and sex for three species of penguins. The data set contains recorded information for 344 penguins: 152 Adelie, 68 Chinstrap, and 124 Gentoo [3]. The data set is attached in the appendix.

The response variable for this data set is the species of penguin, Adelie, Gentoo, and Chinstrap. The predictor variables are the penguins’ body measurements, bill length, bill depth, flipper length, body mas, the sex of the penguin, and the island the penguin was found on.

**Model**

To model our data, we are using three different machine learning operators, Python, R Studio, and MATLAB.  The R Studio code can be found in the appendix.

KNN in Python

KNN is a supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point [8]. Using species as the target variable and bill length and bill depth as the predictor variables, KNN was used to classify the penguins. Two rows were missing most of their information and so those rows were dropped. Because the two predictor variables were both in terms of millimeters, scaling the variables was not necessary. Test sizes from .10 to .50 were tested and a test size of .35 gave the best results. To find the optimal k value k was iterated over from 1 to 40. The best k values were 3, 5 and 7 which yielded an accuracy of 98.3%, which can be seen in Figure 1. Figure 2 shows the boundary plot. Finally, a confusion matrix was created to visualize the model’s accuracy with true label against predicted, which can be found in Figure 3.

Chart, treemap chart

Description automatically generatedChart, map, scatter chart

Description automatically generatedChart

Description automatically generated

Figure 3

Figure 2

Figure 1

Multiple Linear Regression in R Studio

A multiple linear regression classifier is a supervised learning classification algorithm used to classify observations into a discrete set of classes. It is a predictive analysis tool, in which regression is followed by a transformation to return a categorical output and thereby producing a decision boundary [8]. For a multiple linear regression model, there is a singular dependent variable and multiple independent variables. Before starting any coding in R Studio to set up a multiple linear regression classifier on our data set, the data set was cleaned by removing the two rows of data that had ‘NA’ for most of the predictor variables. After doing this, 342 penguins were left, instead of the 344 which were initially recorded.

After setting up the workspace in R Studio, the first step was to make graphs and scatter plots for the different variables in the data set. It should be noted that the sex of the penguin was not used as a variable for this analysis. When comparing the variables against one another, each graph and scatter plot had a positive linear relationship. However, this only occurred when considering a multiple linear regression mode, and not a linear model, as seen in the graphs in Figure 4. Figure 5 shows the pairwise plot matrix overview of the four, body measurement independent variables.

Chart, scatter chart

Description automatically generatedChart

Description automatically generated

Figure 5

Figure 5

Figure 4

On initial look at the data, deciding whether to use island as a dependent variable was thought about. However, after looking at the graphs, it was decided that it would be used, since only Adelie penguins live on Torgersen Island, both Adelie and Chinstrap live on Dream Island, and both Adelie and Gentoo penguins live on Biscoe Island. This ensures a better result when using the test-train model, since the island a penguin lives on depends on the species of penguin. Figure 6 below shows the results of the number of penguins per island.

Chart, bar chart

Description automatically generated

Figure 6

Since some of the variables were characteristic or categorical, they had to be manipulated into a numerical factor variable. Once this was established, a linear regression model was built with a test-train ratio of 80-20. The first regression model built included the independent variables body measurements flipper length, bill length, bill depth, and body mass, and the dependent variable species [5]. The next six models built used body mass as the dependent variable, and the independent variables were all the possible combinations of each of the body measurements [6].

Decision Trees in MATLAB

To create and analyze the decision tree models we used the Classification Learner App in the Statistics and Machine Learning Toolbox in MATLAB. We then exported and used the models created in the MATLAB command window.

The Decision Tree classifier algorithm splits data into progressively specific categories in order to find the best final class. The upsides to decision trees are that they are fast to fit and predict, easy to interpret, and use little memory. A downside is if the tree is overly complex it may overfit. Generally, the data set needed for decision tree models needs to be high-quality to produce good results. This classifier can process numerical and categorical predictors separately or together. Since the Palmer Penguins data set is considered high-quality and has both numerical and categorical data, the decision tree model was a good choice of classifier.

From the original Palmer Penguins data set of 344 x 8, we cleaned it to remove the rows with full Na values resulting in a 342 x 8 data set. We then omitted the year variable in our decision tree models due to the lack of effect they had on the model. We set the response class to species. This means the algorithm used six predictors: island, bill length, bill depth, flipper length, body mass and sex. The algorithm classified into three response classes: Adelie, Chinstrap, and Gentoo. Our train test split for our decision tree models was the standard 80/20. This means we trained 274 rows of data and tested 68 rows of data.

Decision Trees have different model flexibility. MATLAB has three default flexibility levels that correspond to three preset classifier types. First, the coarse tree has low model flexibility with the maximum number of splits set to four. Next, the medium tree has medium flexibility with the maximum number of splits set to twenty. Lastly, the fine tree has high flexibility with the maximum number of splits set to one hundred. In addition to the preset classifier types, MATLAB allows to set hyperparameter options to change the maximum number of splits. After analyzing the data set, we decided to test out five trees with maximum splits being two through six.

In MATLAB, we chose to set up our session using the default five-fold cross validation. Using this method helped us lower the variance in our final results. It is a common method to help prevent data leakage. For the split criterion, deciding when to split the nodes, we used Gini’s diversity index. Gini’s diversity index is the default in MATLAB. It measures node impurity in the range zero to one. If the Gini index is zero, the node is pure meaning that the node has only one class. As the Gini index gets closer to the value of one, the nodes become increasingly less pure with one being a random distribution over the classes.

To visually analyze our five different decision trees, we used a combination of scatter plots, confusion matrices, receiver operating characteristic curves (ROC curves), area under the ROC curve (AUC)  scores, and parallel coordinate plots. We used the accuracy score and total cost for both training and testing results to directly compare the five models.

**Results**

KNN in Python

The final result for the KNN model was an accuracy of 98.3% meaning that 98.3% of the time it predicted the correct type of penguin for the test data. This is shown best in the confusion matrix in Figure 3.

Multiple Linear Regression in R Studio

With the 80-20 test-train model, the multiple regression model suggested that body mass was not that influential and thus could be thought of as another dependent model. Figure 7 shows the console output after running the classification simulation. Hence, the reason for making the six different models with the response variable as body mass.

Text

Description automatically generatedA picture containing text

Description automatically generatedChart, scatter chart

Description automatically generatedAfter running the test on the multiple linear regression model that used the predictor variables flipper length, bill length (culmen length), bill depth (culmen depth), and body mass, the result was that the classification was 94.6% accurate. Figures 9 and 10 show the results of the classification simulation. As stated, since body mass was not that influential and was used as a response variable to make six other models that were all the possible combinations of each of the body measurements, testing the residual vs. the predicted and the QQ plots helped determine that the most important variable in determining the species of penguin is the flipper length.

Figure 9

Figure 8

Figure 7

Decision Trees in MATLAB

There were five decision tree models produced in MATLAB with two to six splits. Two of the models were split the exact same (model 4 and 5). The lowest accuracy score and highest total cost was produced by the two-split model. The highest accuracy score and lowest total cost was produced by the six-split model. The accuracy and total cost scores (below) were the main way we compared the models.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Max Splits | 2 | 3 | 4 | 5 | 6 |
| Accuracy  % (Validation) | 93.1 | 94.5 | 96.0 | 96.0 | 96.7 |
| Total Cost % (Validation) | 19 | 15 | 11 | 11 | 9 |
| Accuracy %  (Test) | 94.1 | 97.1 | 97.1 | 97.1 | 97.1 |
| Total Cost % (Test) | 4 | 2 | 2 | 2 | 2 |

We also considered training time. Since each time was under fifteen seconds, we decided to not include that as a main comparison value. We visualized our results using the following graphs:

Chart, radar chart

Description automatically generatedChart, scatter chart

Description automatically generated

Figure 10

Figure 9

Diagram

Description automatically generatedWord

Description automatically generated with low confidenceChart, line chart

Description automatically generatedChart, radar chart

Description automatically generatedChart, radar chart, line chart

Description automatically generated

Figure 13

Figure 12

Figure 11

Figure 15

Figure 14

**Conclusions**

Using Python was straightforward and easy to work with when doing KNN classification. The results were very accurate, and the process was simple to work with. Completing multiple linear regression analysis on the palmer penguin’s data set was a little complicated, however it did produce a fairly accurate result. We were also able to figure out that the body mass variable has very little to do with the classification process, whereas the flipper length is very important. MATLABs Classification learner made interpreting and making decision tree models straight forward. We found that the six split tree most accurately modeled our data and the two split tree modeled our data the least accurately. With all training times being roughly equivalent, we would choose the six split model to use in practice. The next steps in would be to include using a bigger data set, splitting the data train test data differently, and using other classifiers such as bagged trees.

References

[1] BSD. (2022). *Sklearn.metrics.confusion\_matrix*. scikit learn,

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion\_matrix.html.

[2] BSD. (2022). *Sklearn.model\_selection.train\_test\_split*. scikit learn,

https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.train\_test\_split.html.

[3] DataCamp. (2022). *palmerpenguins*. RDocumentation,

https://www.rdocumentation.org/packages/palmerpenguins/versions/0.1.1.

[4] Harrison, Onel. “Machine Learning Basics with the K-Nearest Neighbors Algorithm.”

*Towards Data Science*, 10 September 2018,

https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-al

gorithm-6a6e71d01761.

[5] K, J. H. (2020, July 10). *Basic data analysis with Palmerpenguins: R-bloggers*. R-bloggers,

https://www.r-bloggers.com/2020/07/basic-data-analysis-with-palmerpenguins/

[6] Logan, N. (2021). *Predicting penguin’s body mass using linear regression models*. RPubs,

https://rpubs.com/nicola\_/825080.

[7] “Machine Learning Classifiers: Definition and 5 Types.” *Indeed*, 14 April 2022,

https://www.indeed.com/career-advice/career-development/classifiers-in-machine-learning.

[8] Mondal, S. (2022, November 30). *Linear vs logistic regression: Linear and logistic*

*regression*. Analytics Vidhya, https://www.analyticsvidhya.com/blog/2020/12/beginners-

take-how-logistic-regression-is-related-to-linear-regression/.

[9] “sklearn.neighbors.KNeighborsClassifier — scikit-learn 1.1.3 documentation.” *Scikit-learn*,

2022, https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.

[10] Pierson, Lillian. *Data Science for Dummies*. 1st ed., For Dummies, 2021.

Appendix

R Studio code:

library(tidyverse)

library(dplyr)

library(corrplot)

library(GGally)

library(gridExtra)

library(sjPlot)

library(ggResidpanel)

penguins<-read\_csv("penguins.csv")

glimpse(penguins)

penguins %>% count(species, name="total")

total\_penguins <- 151+68+123

total\_penguins

penguins\_islands <- penguins %>% count(island, name="penguins")

penguins\_islands

flipper\_mass <- ggplot(data = penguins, aes(x = flipper\_length\_mm, y = body\_mass\_g)) +

geom\_point(aes(color = species, shape = species), size = 3, alpha = 0.8) +

labs(title = "Flipper Length and Body Mass", x = "Flipper Length (mm)", y = "Body Mass (g)",

color = "Penguin Species", shape = "Penguin Species")

flipper\_mass

flipper\_bill <- ggplot(data = penguins, aes(x = flipper\_length\_mm, y = bill\_length\_mm)) +

geom\_point(aes(color = species, shape = species), size = 3, alpha = 0.8) +

labs(title = "Flipper Length and Bill Length", x = "Flipper Length (mm)", y = "Bill Length (mm)",

color = "Penguin Species", shape = "Penguin Species")

flipper\_bill

bill\_len\_dep <- ggplot(data = penguins, aes(x = bill\_length\_mm, y = bill\_depth\_mm, group = species)) +

geom\_point(aes(color = species, shape = species), size = 3, alpha = 0.8) +

geom\_smooth(method = "lm", se = FALSE, aes(color = species)) +

labs(title = "Bill Length and Bill Depth", x = "Bill Length (mm)", y = "Bill Depth (mm)",

color = "Penguin species", shape = "Penguin species")

bill\_len\_dep

bill\_no\_species <- ggplot(data = penguins, aes(x = bill\_length\_mm, y = bill\_depth\_mm)) +

geom\_point() +

labs(title = "Bill Dimensions (Omit Species)", x = "Bill Length (mm)", y = "Bill Depth (mm)") +

geom\_smooth(method = "lm", se = FALSE, color = "gray50")

bill\_no\_species

penguins\_per\_island <- ggplot(penguins)+

geom\_point(aes(island, species, color=species, shape=species))+

labs(title="Penguin Species Per Island")

penguins\_per\_island

hbarpenguins <- ggplot(penguins) +

geom\_bar(aes(island, fill=species)) +

coord\_flip() +

labs(title="Types of Penguins per Island", x="Island", y="Amount of Penguins")

hbarpenguins

penguinshisto <- ggplot(penguins) +

geom\_histogram(aes(flipper\_length\_mm, fill=species), position='identity', alpha=0.5) +

labs(title="Frequency of Penguin Flipper Lengths per Species", x="Length (mm)", y="Frequency")

penguinshisto

size\_per\_penguin <- ggplot(penguins) +

geom\_point(aes(flipper\_length\_mm, body\_mass\_g, color=species, shape=species)) +

labs(title="Body Mass vs. Flipper Length")

size\_per\_penguin

species\_per\_island <- ggplot(data = penguins) +

geom\_point(mapping = aes(x=flipper\_length\_mm, y = body\_mass\_g, color = species))+

labs(title="Body Mass vs. Flipper Length")+

geom\_smooth(aes(x=flipper\_length\_mm, y = body\_mass\_g)) +

facet\_wrap(~island)

species\_per\_island

penguins %>% select(species, body\_mass\_g, ends\_with("\_mm")) %>% GGally::ggpairs(aes(color = species),

columns = c("flipper\_length\_mm", "body\_mass\_g", "bill\_length\_mm", "bill\_depth\_mm"))

penguins$species <- as.factor(penguins$species)

penguins$island <- as.factor(penguins$island)

penguins$sex <- as.factor(penguins$sex)

set.seed(1234)

index <- sample(1:nrow(penguins), size = nrow(penguins)/2)

speciesnumeric <- as.numeric(penguins$species)

penguins$speciesnumeric <- speciesnumeric

train <- penguins[index,]

test <- penguins[-index,]

fit <- lm(speciesnumeric ~ flipper\_length\_mm + bill\_length\_mm + bill\_depth\_mm + body\_mass\_g, train)

summary(fit)

predRes <- round(predict(fit, test))

predRes[which(predRes>3)] <- 3

predRes <- sort(names(pCol))[predRes]

test$predRes <- predRes

ggplot(test, aes(x = species, y = predRes, color = species)) + geom\_jitter(size = 3)

table(test$predRes, test$species)

model1 <- lm(body\_mass\_g ~ bill\_length\_mm, penguins)

summary(model1)

model2 <- lm(body\_mass\_g ~ bill\_depth\_mm, penguins)

summary(model2)

model3 <- lm(body\_mass\_g ~ flipper\_length\_mm, penguins)

summary(model3)

model4 <- lm(body\_mass\_g ~ bill\_length\_mm + bill\_depth\_mm, penguins)

summary(model4)

model5 <- lm(body\_mass\_g ~ bill\_length\_mm + flipper\_length\_mm, penguins)

summary(model5)

model6 <- lm(body\_mass\_g ~ bill\_length\_mm + bill\_depth\_mm + flipper\_length\_mm, penguins)

summary(model6)

tab\_model(model1, model2, model3, model4, model5, model6, dv.labels = c("Model 1", "Model 2", "Model 3", "Model 4", "Model 5", "Model 6"))

resid\_panel(model1,plots=c("resid","qq"))

resid\_panel(model2,plots=c("resid","qq"))

resid\_panel(model3,plots=c("resid","qq"))

resid\_panel(model4,plots=c("resid","qq"))

resid\_panel(model5,plots=c("resid","qq"))

resid\_panel(model6,plots=c("resid","qq"))