C964: Computer Science Capstone

Task 2 Parts A, B, C, and D

Hiep Pham

ID #001467696

WGU Email: hpham25@wgu.edu

Date: 07/21/2023

Table of Contents

Part A: Project Proposal for Business Executives

Letter of Transmittal

Hiep Pham

ML Creatives, Founder

246 Apple St.

Denver, CO

Han Solo

Denver Zoo, President

357 Animal St.

Denver, CO

Data Product Proposal to Identify Penguin Species

Greetings Mr. Solo,

With the Antarctic ice sheet completely melting, your zoo has mass imported Adélie, Gentoo, and Chinstrap penguins from Antarctica. From our previous conversations, I understand that large amounts of penguins have caused identification issues, as colonies intertwine. The close resemblance of these rookeries has strained the zoologists at your organization in the task of identifying the correct penguin species to ensure proper segregation and organization.

ML Creatives is proposing a data product that may be of use to your organization. Our product, let’s call it the Penguin Identifier, will aid your zoologists in identifying the penguin species simply by inputting the penguin’s measurements. The Penguin Identifier will output a species, with a great degree of confidence, and will help your organization streamline the processing of mass amounts of penguins.

The estimated total cost will be $50,000. I understand this matter is of the highest urgency for your organization, and I can guarantee a 1 week's return time for a completed product. Despite my lack of knowledge about penguins, we are a seasoned software development team and have experience in developing machine-learning models. I am confident we are the solution to your predicament.

Looking forward to hearing back,

Hiep Pham

ML Creatives, Founder

Project Recommendation

Problem Summary

The Penguin Identifier will identify Adelie, Gentoo, or Chinstrap depending upon the provided measurements of culmen length, culmen depth, flipper length, and body mass. The product will be developed as a command-line interface program, easily portable on any computer with network access to the dataset we will be using to train our Machine Learning model.

The Penguin Identifier will expedite the processing of the penguins into your zoo. I understand with the mass amounts of penguins, time is of the utmost importance. If the penguins are left to intermingle and mix colonies, there may be violence and wars between the penguin colonies. Understanding the urgency, we will provide a finished product within 1 week to separate the penguins accordingly.

The Penguin Identifier meets your needs by enabling a streamlined process for tagging penguin species. Though you don't have to follow these suggestions, I picture a line of penguins getting measured, identified using the Penguin Identifier, and tagged or placed in the correct colony. This will enable your zoo to efficiently identify penguins rather than doing guesswork and relying on varying zoologist experiences.

The Penguin Identifier will be delivered as a command-line interface program, functioning on any computer with Python installed with network access. The penguin identifier will achieve an accuracy rate of 90%+, inspiring confidence in species identification. The program will enable a streamlined process for tagging and separating the penguins.

Application Benefits

The Penguin Identifier directly meets the Denver Zoo's pressing needs by enabling a solution to effectively identify and segregate the mass influx of Adelie, Gentoo, and Chinstrap penguins from Antarctica. The challenges caused by intertwined colonies are alleviated by our data product, which leverages machine learning to accurately predict the penguin species based on provided measurements. This application enables a streamlined approach to the identification process, reducing the reliance on zoologist/zookeeper expertise and relieving the burden upon the most experienced personnel. Most importantly, the Penguin Identifier responds to the Denver Zoo's number 1 priority: maintaining the health of the penguins.

Application Description

The Penguin Identifier is the data product created to address the Denver Zoo's penguin species identification challenges. The data product utilizes a machine learning approach, specifically through Logistic Regression. The application will use Logistic Regression to build a prediction model to accurately identify a penguin species based on culmen length, culmen depth, flipper length, and body mass. When a user inputs this information, the Penguin Identifier will return the penguin species with a 90% or higher confidence level. The identified penguin species will allow the zoologist/zookeeper to categorize the penguins, ensuring their health by preventing territorial/colonial conflicts.

Data Description

The raw data used in the Penguin Identifier originates from the "Palmer Archipelago (Antarctica) penguin data", created by Parul Pandey on www.kaggle.com (Gorman KB). This data contains the penguin species, the island where the penguin originates from, culmen length, culmen depth, flipper length, body mass, and the penguin sex.

The data type is a mixture of nominal and quantitative, containing both qualitative descriptors of the penguin (island and sex), and numerical descriptors of the penguin (culmen length, culmen depth, flipper length, and body mass. The data structure is like a table, where there are rows of values, corresponding to columns of value type. However, the island column will be omitted during data processing because the scenario dictates that island origination will be unknown to the zookeepers and zoologists. Also, the sex column will be omitted, as the culmen measurements alone can determine the sex & species (Urton). This structure is stored as a comma-separated values (CSV) file.

The dependent variable will be the penguin species, and the independent variable will be the penguin dimensions (culmen length, culmen depth, flipper length, and body mass). The independent variables (penguin dimensions) will be used to predict the dependent variable (penguin species).

The primary limiter in this program is the dataset size. At 342 entries, this data size is small compared to other machine learning models that have thousands of entries to train their model. The data will be parsed and processed for any anomalies, outliers, and incomplete data. If the anomalies would affect the prediction model meaningfully, the data will be omitted to ensure correct training occurs. Thankfully, we don’t expect penguins to evolve at a rapid pace in the next few years, so little to no maintenance is required for the dataset.

Objectives and Hypothesis

The objective of the Penguin Identifier program is to provide the Denver Zoo with a reliable solution to identify and segregate the mass import of Adelie, Gentoo, and Chinstrap penguins based on their measurements. The desired outcome of implementing the Penguin Identifier is the avoidance of territorial and colonial conflicts between the intertwined penguin species, maintaining the safety and health of the penguins and zookeepers alike.

The hypothesis is: Provided a reliable dataset, the Penguin Identifier, utilizing a Logistic Regression model, will accurately predict penguin species with a confidence level of 90% or greater.

Methodology

Our application development will adopt the Waterfall approach. There are two primary reasons for this choice:

- We do not expect to work closely with our customers, which would've called for an Agile approach.
- We fully understand the requirements and scope of this application, enabling us to work through the phases without needing to revisit previous phases.

The waterfall methodology comprises of these 5 phases: Requirements, Design, Implementation, Verification/Testing, and Deployment & Maintenance. We will discuss the details of each phase below:

1. REQUIREMENTS

The requirements phase is the most important in the Waterfall methodology. Here, we will scope and define our application's requirements concerning the client's needs. This is the phase where we determine which species of penguins are needed and the best method to identify their species. This phase is where we expect to discuss with the client the most, gather their requirements, and specifications, and define constraints and restraints. Notably, we understand the Zoo requires a basic program that's reliable and efficient. Hence, we continue onto phase 2, Design.

2. DESIGN

In the Design phase, we utilize the requirements gathered from the Requirements phase to create the design of system architectures, databases, and user interfaces. Because we understand the client requires a basic program, we are going to design the product to be interacted with through the command line interface. To ensure consistent behavior, we will specify that the program will be run using the latest Python and PyCharm Community Edition IDE versions. Due to the urgent nature of the situation, the program is designed to be minimal in features, and its only job will be to identify penguins. We also establish the dataset we want to use in this phase, aligning it with the requirements. Because the design is now complete, we can move on to phase 3: Implementation.

3. IMPLEMENTATION

In Implementation, our team of developers will begin coding the application based on the requirements and design blueprint. As noted in the Design phase, the developers will be writing the source code in PyCharm Community Edition, interpreted by Python. The developers are developing this program on a Windows Operating System, understanding that the Denver Zoo primarily uses Windows OS for its IT infrastructure. The developers will use modules to parse the dataset and build the Logistic Regression prediction model. This model will be trained using created training data. Testing data will also be created, useful for phase 4: Testing. Additionally, we will also be using a plotting module to create a scatter matrix, showing the user the relations between penguin measurements. Here, we can expect that they will be positively correlated, such as flipper length growing along with body mass. Using another module, we will display these graphs for the user’s situational awareness of the application startup. We expect this phase to be the quickest, because of the clearly defined requirements and design.

4. VERIFICATION/TESTING

In Verification/Testing, we ensure that the prediction model we used is reliable in predicting penguin species. We create a confusion matrix, which will display all correctly predicted penguins on the diagonal line, and wrongly predicted penguins in all other locations. The confusion matrix will assure the user that the prediction model can accurately predict penguin species because it has reserved some of the datasets as test data. The prediction model will use the remaining dataset to train the model. As detailed by our requirements, we’re requiring a 90% accuracy rate, and at any point, if the accuracy rate falls below 90%, we will have to return to the beginning of the waterfall methodology to seek a solution. Through our testing, we have yet to see the accuracy rate fall below 90%. This signals that we can confidently move to the last phase: Deployment & Maintenance.

5. DEPLOYMENT & MAINTENANCE

The deployment & maintenance phase is the last phase of the waterfall methodology. The application will be deployed as a Python file, meant to be run on a PyCharm IDE with Python installed on the computer. The application will require the installation of “pandas”, “scikit-learn”, and “matplotlib” modules on the IDE. Lastly, the workstation will require an internet connection to pull the dataset from the GitHub repository. Due to the nature of the situation, we do not expect the client to utilize this application long-term. Maintenance for this application will be minimal to null, because of its simplicity and intentional lack of features. If the client wishes to continue using this application with additional penguin species, it is recommended to contact us or another software development team to expand the application’s capabilities.

Funding Requirements

The project will require funding of $50,000 all-inclusive, with a return time of 1 week. This funding will solely be for personnel costs, because of the 1 week return time constraint. We understand you require a finished product that performs all required tasks accurately, and the funding will be used to compensate our best software developers. It is expected that they may have to work overtime throughout the week to accomplish this product to meet the deadline time. The expected cost for the environment is $0 because the software developers will be working remotely. Licensing cost will be $0 because we expect to use a public domain-licensed dataset. The software developers will be using PyCharm Community Edition, so the cost will also be $0.

Data Precautions

The data we will use will contain 0 sensitive or protected data because it will only relate to penguins. As such, no additional time will be required to review privacy general guidelines. We plan to use a public domain-licensed dataset from www.kaggle.com, so no additional precautions are required.

Developer’s Expertise

The developers have their B.S. Computer Science and have experience working on AI and Machine Learning through the WGU courses. The developers also developed extensive Python experience through developing data structures and their capstone, which is focused on machine learning. This expertise will be crucial in the development of this machine-learning data product.

Part B: Project Proposal

Problem Statement

The Denver Zoo faces a critical challenge following the melting of the Antarctica polar ice caps. The Zoo has received a mass import of the Adelie, Gentoo, and Chinstrap penguins. To appropriately care for the penguins, the zoo needs to categorize and separate the penguins according to their species to prevent colonial wars. The zoo requires a solution that efficiently identifies a penguin's species depending on its dimensions.

Customer Summary

The client is the Denver Zoo, an internationally recognized zoological facility known for its dedication to wildlife conservation and care. Specifically, the customer is the IT leadership department of the Denver Zoo, tasked with finding a solution that will resolve its challenge.

The proposed machine learning application will successfully address the Denver Zoo's challenge of quickly and accurately identifying penguin species. By using a Logistic Regression, the Penguin Identifier will predict the most likely species of penguin based on the inputted penguin dimensions. The Penguin Identifier predicts in near real-time, ensuring an efficient and quick identification process. The speed of the prediction enables the zoo to properly categorize and care for the penguins at an efficient rate.

Existing System Analysis

The Denver Zoo has no process for efficiently identifying penguins currently. The situation of the polar ice caps is unprecedented, and the Zoo has never needed to quickly identify penguins before. Currently, the Zoo still relies on the zoologists to make their best guess as to what the penguin species is, based on experience.

Unfortunately, zoologists' and zookeepers' experiences dramatically vary. Zookeeper Kelly can say the penguin looks most like an Adelie, but Zookeeper Brad can also claim the penguin is a Gentoo because he did a study on them in graduate school. Both must then look online at pictures, consult a zoologist, and make an educated guess. This process is inefficient and frankly unreliable. The Penguin Identifier will use machine learning to accurately and quickly identify penguins based on inputted penguin measurements. The confidence level will be displayed at every application start. The penguin identification process will be made dramatically more efficient, enabling the Denver Zoo to care for the penguins appropriately.

Data

The raw data set will be sourced from "Palmer Archipelago (Antarctica) penguin data" by Parul Pandey on www.kaggle.com (Gorman KB). Specifically, we will be using the data from the "penguins_size.csv" file.

The data will be collected by downloading from www.kaggle.com. Because the data has a CC0: Public Domain license, we can be assured the data is free and not copyrighted. The data will then be parsed and processed for any anomalies, outliers, and incomplete data. If the anomalies would affect the prediction model meaningfully, the data will be omitted to ensure correct training occurs. Thankfully, we don't expect penguins to evolve at a rapid pace in the next few years, so little to no maintenance is required for the dataset.

Project Methodology

Our application development will adopt the Waterfall approach. There are two primary reasons for this choice:

- We do not expect to work closely with our customers, which would’ve called for an Agile approach.
- We fully understand the requirements and scope of this application, enabling us to work through the phases without needing to revisit previous phases.

The waterfall methodology comprises of these 5 phases: Requirements, Design, Implementation, Verification/Testing, and Deployment & Maintenance. We will discuss the details of each phase below:

1. REQUIREMENTS

The requirements phase is the most important in the Waterfall methodology. Here, we will scope and define our application's requirements concerning the client's needs. This is the phase where we determine which species of penguins are needed and the best method to identify their species. We will find the correct dataset corresponding to the requirements, and tailor the data to within the scope of the requirements. This phase is where we expect to discuss with the client the most, gather their requirements, and specifications, and define constraints and restraints. Notably, we understand the Zoo requires a basic program that’s reliable and efficient. Hence, we continue onto phase 2, Design.

2. DESIGN

In the Design phase, we utilize the requirements gathered from the Requirements phase to create the design of system architectures, databases, and user interfaces. Because we understand the client requires a basic program, we are going to design the product to be interacted with through the command line interface. To ensure consistent behavior, we will specify that the program will be run using the latest Python and PyCharm Community Edition IDE versions. The program itself will be created similarly, using Python 3.11 and in PyCharm Community Edition version 2023.1.3. PyCharm enables quick module installation, enabling the graphical features of the design. Due to the urgent nature of the situation, the program is designed to be minimal in features, and its only job will be to identify penguins. We also establish the dataset we want to use in this phase, aligning it with the requirements. Because the design is now complete, we can move on to phase 3: Implementation.

3. IMPLEMENTATION

In Implementation, our team of developers will begin coding the application based on the requirements and design blueprint. As noted in the Design phase, the developers will be writing the source code in PyCharm Community Edition version 2023.1.3, interpreted by Python version 3.11. The developers are developing this program on a Windows Operating System, understanding that the Denver Zoo primarily uses Windows OS for its IT infrastructure. The developers will use the “pandas” version 2.0.3 module to parse the dataset, and “scikit-learn” version 1.3 to build the Logistic Regression prediction model. This model will be trained using data created from the train_test_split function from “scikit-learn”. Train_test_split will create training and testing data, useful for phase 4: Testing. Additionally, we will also be using the “pandas” plotting module to create a scatter matrix, showing the user the relations between penguin measurements. Here, we can expect that they will be positively correlated, such as flipper length growing along with body mass. Using “matplotlib” version 3.7.2, we will display these graphs for the user’s situational awareness of application startup. We expect this phase to be the quickest, because of the clearly defined requirements and design.

4. VERIFICATION/TESTING

In Verification/Testing, we ensure that the prediction model we used is reliable in predicting penguin species. We create a confusion matrix, from the “scikit-learn” metrics module. The confusion matrix will display all correctly predicted penguins on the diagonal line, and wrongly predicted penguins in all other locations. The confusion matrix will assure the user that the prediction model can accurately predict penguin species because it has reserved 20% of the dataset as test data. The prediction model will use the remaining 80% of the dataset to train the model. As detailed by our requirements, we're requiring a 90% accuracy rate, and at any point, if the accuracy rate falls below 90%, we will have to return to the beginning of the waterfall methodology to seek a solution. Through our testing, we have yet to see the accuracy rate fall below 90%. This signals that we can confidently move to the last phase: Deployment & Maintenance.

5. DEPLOYMENT & MAINTENANCE

The deployment & maintenance phase is the last phase of the waterfall methodology. The application will be deployed as a Python file, meant to be run on a PyCharm IDE with Python installed on the computer. The application will require the installation of "pandas", "scikit-learn", and "matplotlib" modules on the IDE. Lastly, the workstation will require an internet connection to pull the dataset from the GitHub repository. Due to the nature of the situation, we do not expect the client to utilize this application long-term. Maintenance for this application will be minimal to null, because of its simplicity and intentional lack of features. If the client wishes to continue using this application with additional penguin species, it is recommended to contact us or another software development team to expand the application's capabilities.

Project Outcomes

The data product will be delivered as a Python file, meant to be run on a PyCharm IDE with Python installed on the computer. A user guide will also be attached, explaining how to get the application up and running. The user guide is detailed below:

INSTALLATION

1. Install the latest version of Python at https://www.python.org/downloads/. Once on the page, click the “Download” button in the center of the page. Follow the directions for other Operating Systems if using an OS other than Windows. Follow the instructions from the installation package and complete the installation.

2. Install the latest version of PyCharm Community Edition at https://www.jetbrains.com/pycharm/download/. Once on the page, scroll down until you find “PyCharm Community Edition”, and click “Download”. Follow the installations from the installation package and complete the installation.

3. Open PyCharm

4. Search for the menu “Open”. This is where we will open the Penguin Identifier program.

5. Navigate to the directory where you have saved the data product. The data product should be a folder named “964HiepPham”, or whatever you changed the name to when you downloaded it.

6. Highlight the package and click “Open” in the window.

7. The project has now opened, but we are not done. We need to install modules to ensure the program runs correctly. Click on “File” in the top left corner.

8. Click the drill down on “Project: 964HiepPham”

9. Click on Python Interpreter. This is where we will install the required modules to run the program.

10. Within the Python Interpreter menu, click on the “+”

11. In the search, type “pandas”, and install the exact module highlighted by clicking “Install Package” at the bottom. The installation may take a while, depending on computer specifications. You can see the progress at the bottom of the PyCharm IDE or wait for the green bar at the bottom of the “Available Packages” menu that says “Package ‘XYZ’ installed successfully”.

12. In the search, type “matplotlib”, and install the exact module highlighted by clicking “Install Package” at the bottom. The installation may take a while, depending on computer specifications. You can see the progress at the bottom of the PyCharm IDE or wait for the green bar at the bottom of the “Available Packages” menu that says “Package ‘XYZ’ installed successfully”.

13. In the search, type “scikit-learn”, and install the exact module highlighted by clicking “Install Package” at the bottom. The installation may take a while, depending on computer specifications. You can see the progress at the bottom of the PyCharm IDE or wait for the green bar at the bottom of the “Available Packages” menu that says “Package ‘XYZ’ installed successfully”.

14. Once all 3 modules have been installed, the INSTALLATION part is complete! Please continue to “USING THE APPLICATION”

USING THE APPLICATION

1. Ensure you do not modify the code at any point.

2. Click on the green arrow on the top right of the PyCharm IDE. This will run the application.

3. A console window will appear from the bottom of your screen showing output. It is recommended you drag the top of the console window up to enlarge the console, and to prevent code modification.

4. The console will display exactly what the code is doing. If you would like to know exactly what the code has done, we recommend enlarging the console and reading the progress steps.

5. Three plots will appear on your screen. This is for your situational awareness. Figure 1 shows the count of each species in the dataset. Figure 2 displays the scatter matrix showing the relationship between the penguin dimensions. Figure 3 displays the Confidence Matrix, showing how accurate the prediction model is (numbers on the diagonal means the model predicted correctly).

6. To continue with the application, close all three plots. The console also directs you to close all graphs.

7. The Penguin Identifier will also show you the confidence level for the prediction model, ranging from 0.0 to 1.0. If at any point, you see the model output a confidence level of less than the agreed-upon .90, please contact us immediately. Though, in our hundreds of test runs, we've never seen the confidence level dip below .90.

8. The program will ask you to enter '1' to continue. Please continue only if you are satisfied with the confidence level or want to continue identifying penguin species. Otherwise, enter '2' to quit the program.

9. The Penguin Identifier instructs you to provide the following measurements in order separated by a space: culmen length in millimeters, culmen depth in millimeters, flipper length in millimeters, and body mass in grams. The console also displays an example and the corresponding prediction.

10. The Penguin Identifier will output the predicted penguin species based on the inputted dimensions.

11. The Penguin Identifier will loop back and ask you again to (1) continue, or (2) quit.

12. Once you have completed identifying all your penguins, you may enter '2' to quit. The program is complete, and you may close out PyCharm.

13. Last reminder, please do not modify the code.

Implementation Plan

The Implementation Plan will follow the detailed waterfall methodology as detailed in the "Project Methodology" section above. In short, ML Creatives will acquire requirements from the Denver Zoo. This will be the most important phase since it's the only phase ML Creatives will be in constant communication with the Denver Zoo. Then, ML Creatives will design, implement, verify & test the product. Lastly, ML Creatives will deploy the application to the Denver Zoo.

In the deployment, ML Creatives will also supply Denver Zoo with the User Guide. The User Guide will contain all information relating to the installation and dependencies of the application. In short, the Denver Zoo will need to install Python, PyCharm Community Edition IDE, and 3 modules within the PyCharm Community Edition IDE.

ML Creatives will integrate a way to test the prediction model on every start-up. The application will output the confidence level of the prediction model using test data segmented from the dataset. The user will not need to test anything once the application is complete, since the testing portion will be completed by ML Creatives. If the Denver Zoo requires expansions upon the application, please reach back out to ML Creatives or another software development team. The dataset is public so Denver Zoo may repurpose the data from www.kaggle.com, but the code itself is not to be distributed before discussions with ML Creatives.

Evaluation Plan

- Describe the verification method(s) to be used at each stage of development.

We will follow the waterfall methodology as noted in the section "Project Methodology". Here, we will describe the verification methods to validate the progression to the next phase:

1. REQUIREMENTS

We can verify the completion of the requirements stage once both ML Creatives and Denver Zoo agree upon the requirements and scope of the data product. Once all requirements and scope have been defined, ML Creatives will move to the design stage to begin developing the agreed-upon application.

2. DESIGN

We can verify the completion of the design stage once the design of system architectures, databases, and user interface are well understood. Because return time is prioritized over a polished application, ML Creatives understand the application will meet the requirements as a command-line interface. In Design, ML Creatives will also determine the appropriate dataset that will meet requirements. Once all designs and datasets have been determined, ML Creatives will move to the implementation stage to code the application.

3. IMPLEMENTATION

We can verify the completion of the implementation stage once the written code can predict a penguin species based on inputted dimensions. Throughout the implementation phase, ML Creatives will also continually check requirements are being met through the application (notably the accuracy score). Once the code is complete and running, ML Creatives will move to the Verify & Test stage to formally validate the program meets all requirements.

4. VERIFICATION & TESTING

We can verify the completion of the verification & testing stage once the program consistently meets the agreed-upon accuracy, of 90%. Throughout the implementation and verification stages, if the program displays an accuracy of below 90% once, ML Creatives will relink with Denver Zoo to revisit the requirements and work internally to redesign the product to ensure requirements are met. Once verification and testing are complete and satisfied, ML Creatives will move to the Deployment stage to officially give Denver Zoo the data product.

5. DEPLOYMENT

We can verify the completion of the deployment stage once Denver Zoo understands and assumes all responsibilities for the application. When Denver Zoo formally agrees to the product as detailed in the requirements, and understands how to use the application, the Deployment stage is officially completed and Product Delivery is validated.

Resources and Costs

Hardware/software cost • All hardware and software are already in possession to develop the data product. The developers will use personal devices and a free IDE to develop the product. No additional hardware and software will require purchasing for the development of this data product.	$0
License cost • The data will be sourced from www.kaggle.com, and a public domain-licensed dataset will be used.	$0
Personnel cost • The cost for the team of developers will be $50,000 in labor, assuming there will be overtime work. ML Creatives require 1 week to complete the product and present it to Denver Zoo.	$50,000 1 week
Deployment, hosting, and maintenance cost • There will be no cost for deployment, hosting, and maintenance cost. The data product will be run locally on a Denver Zoo computer. There will be no required maintenance.	$0

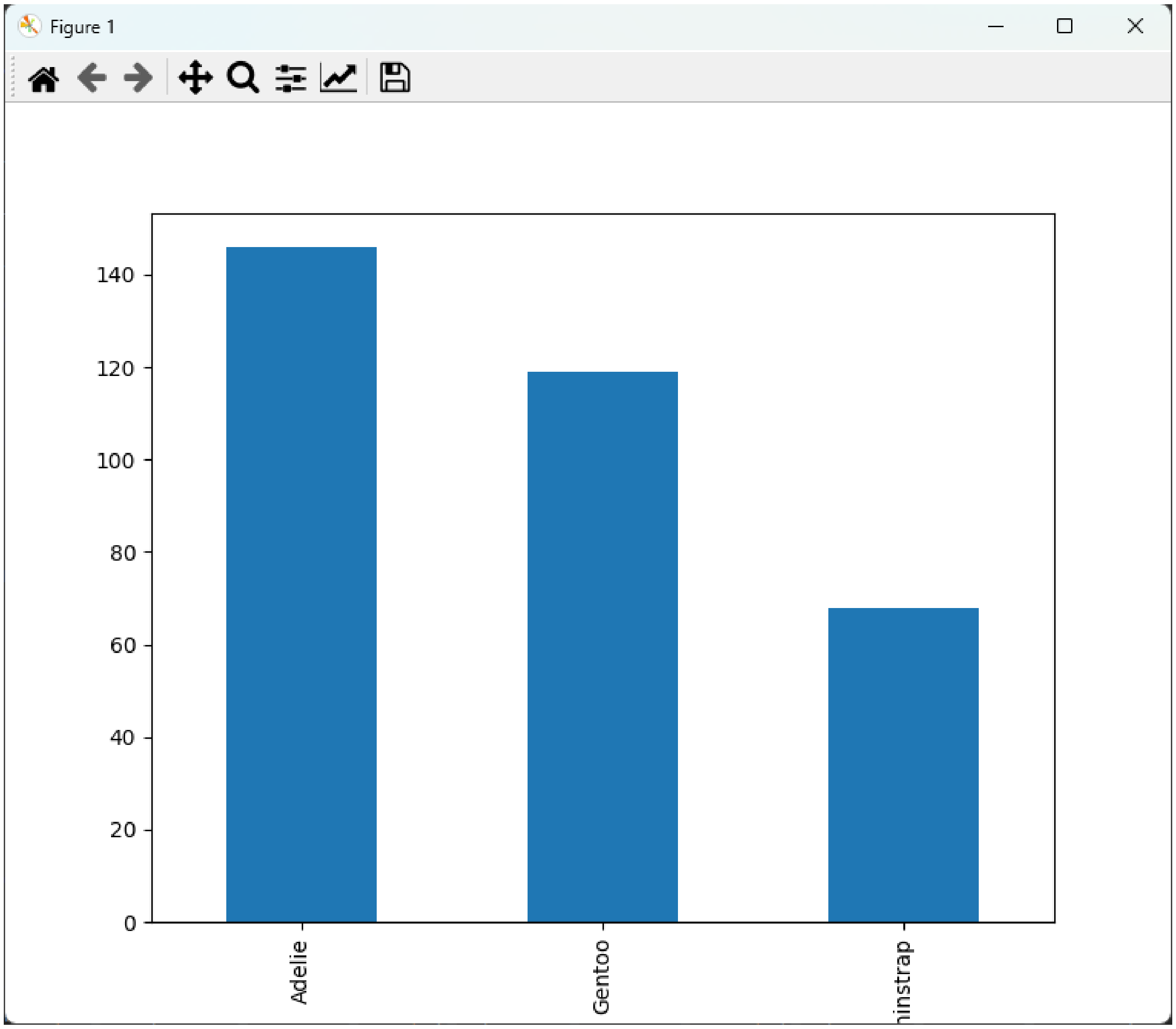
Timeline and Milestones

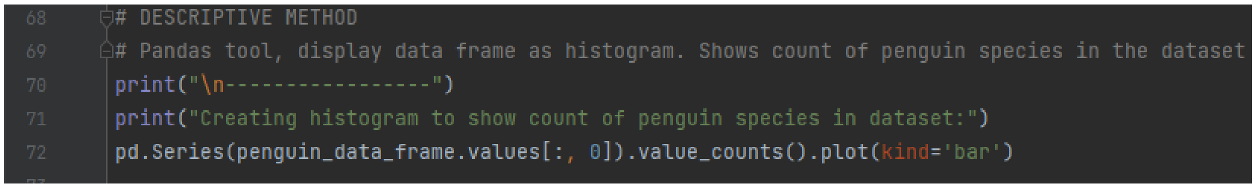
The timeline and milestone will reflect the “Product Methodology” section. We will use the Waterfall Methodology

Objective	**Start**	**End**
Define Requirements	08/01/2023	08/02/2023
Complete Product Design	08/03/2023	08/03/2023
Implement Product code	08/04/2023	08/04/2023
Verify & Test Application	08/05/2023	08/06/2023
Deploy the application to Denver Zoo	08/07/2023	08/07/2023

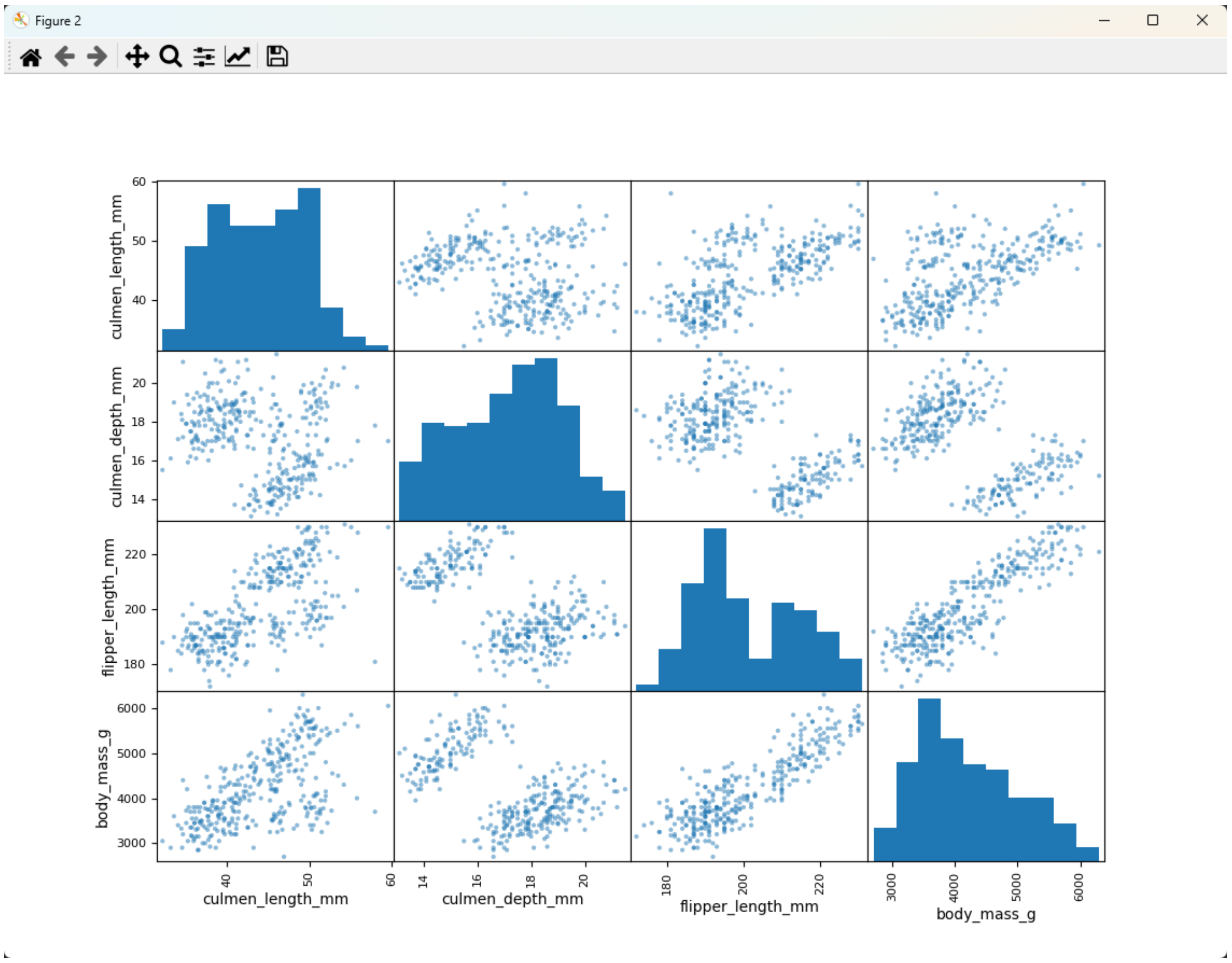
Part C: Application

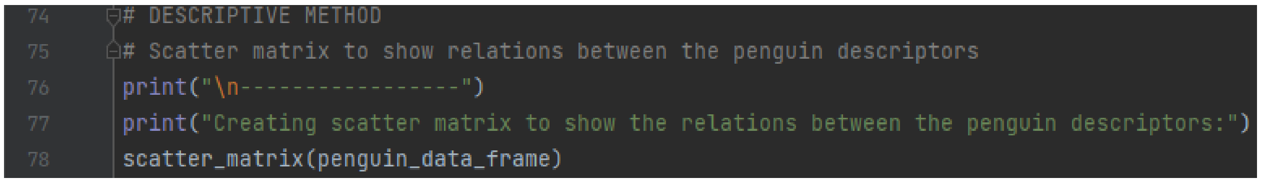
- **Three visualizations (images). Static images are permissible.**
- **A *Descriptive method* = anything that describes the data. Total samples of each penguin species from the dataset**



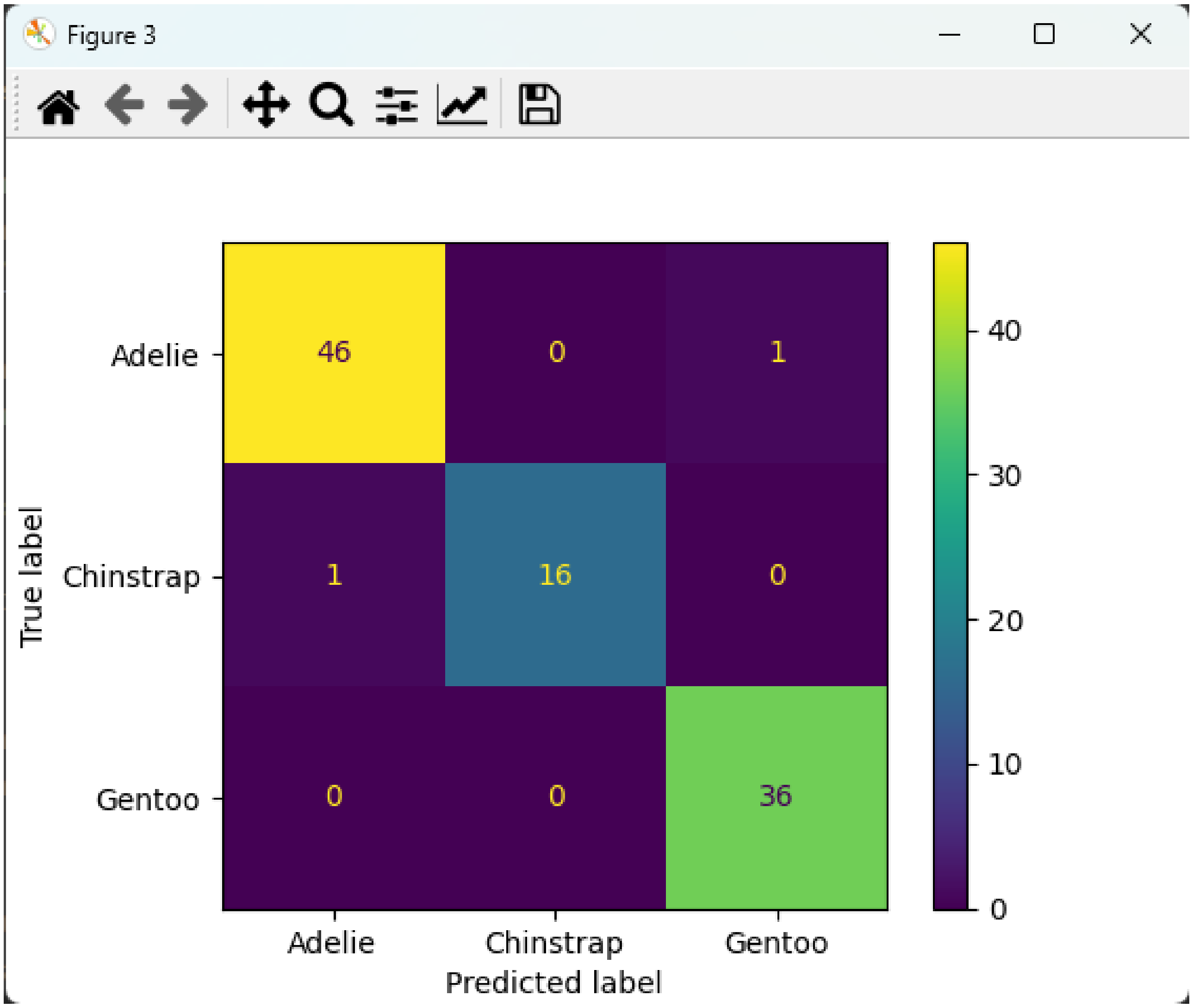


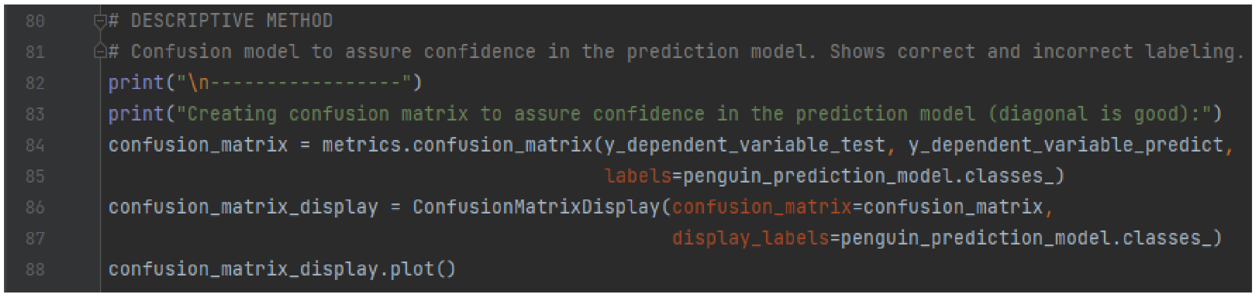
```
# DESCRIPTIVE METHOD
# Pandas tool, display data frame as histogram. Shows count of penguin species in the dataset
print("\n----------------")
print("Creating histogram to show count of penguin species in dataset:")
pd.Series(penguin_data_frame.values[:, 0]).value_counts().plot(kind='bar')
```





```
# DESCRIPTIVE METHOD
# Scatter matrix to show relations between the penguin descriptors
print("\n----------------")
print("Creating scatter matrix to show the relations between the penguin descriptors:")
scatter_matrix(penguin_data_frame)
```

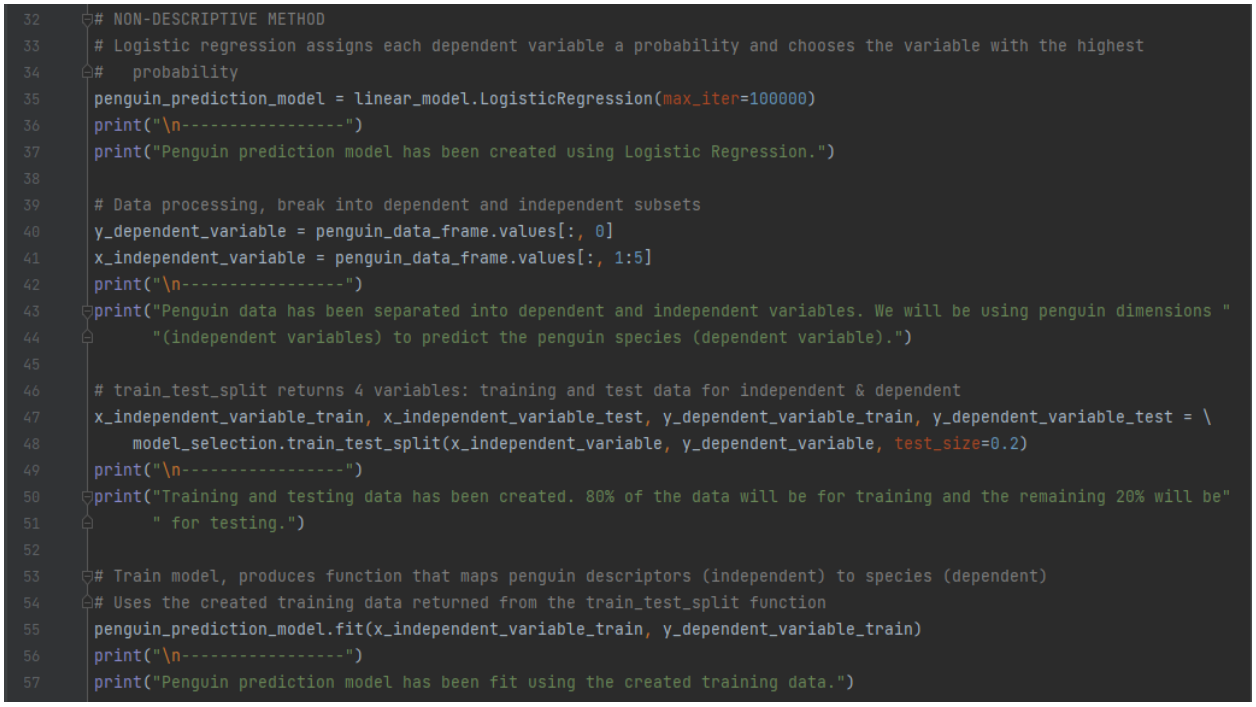




```
# DESCRIPTIVE METHOD
# Confusion model to assure confidence in the prediction model. Shows correct and incorrect labeling.
print("\n-----------------")
print("Creating confusion matrix to assure confidence in the prediction model (diagonal is good):")
confusion_matrix = metrics.confusion_matrix(y_dependent_variable_test, y_dependent_variable_predict,
                                            labels=penguin_prediction_model.classes_)
confusion_matrix_display = ConfusionMatrixDisplay(confusion_matrix=confusion_matrix,
                                                  display_labels=penguin_prediction_model.classes_)
confusion_matrix_display.plot()
```

- **A *Non-descriptive method* = anything that infers from the data, i.e., makes predictions or prescriptions.**
- **An application of "machine learning" in the non-descriptive OR descriptive method (most data analysis algorithms are acceptable -including regression).**

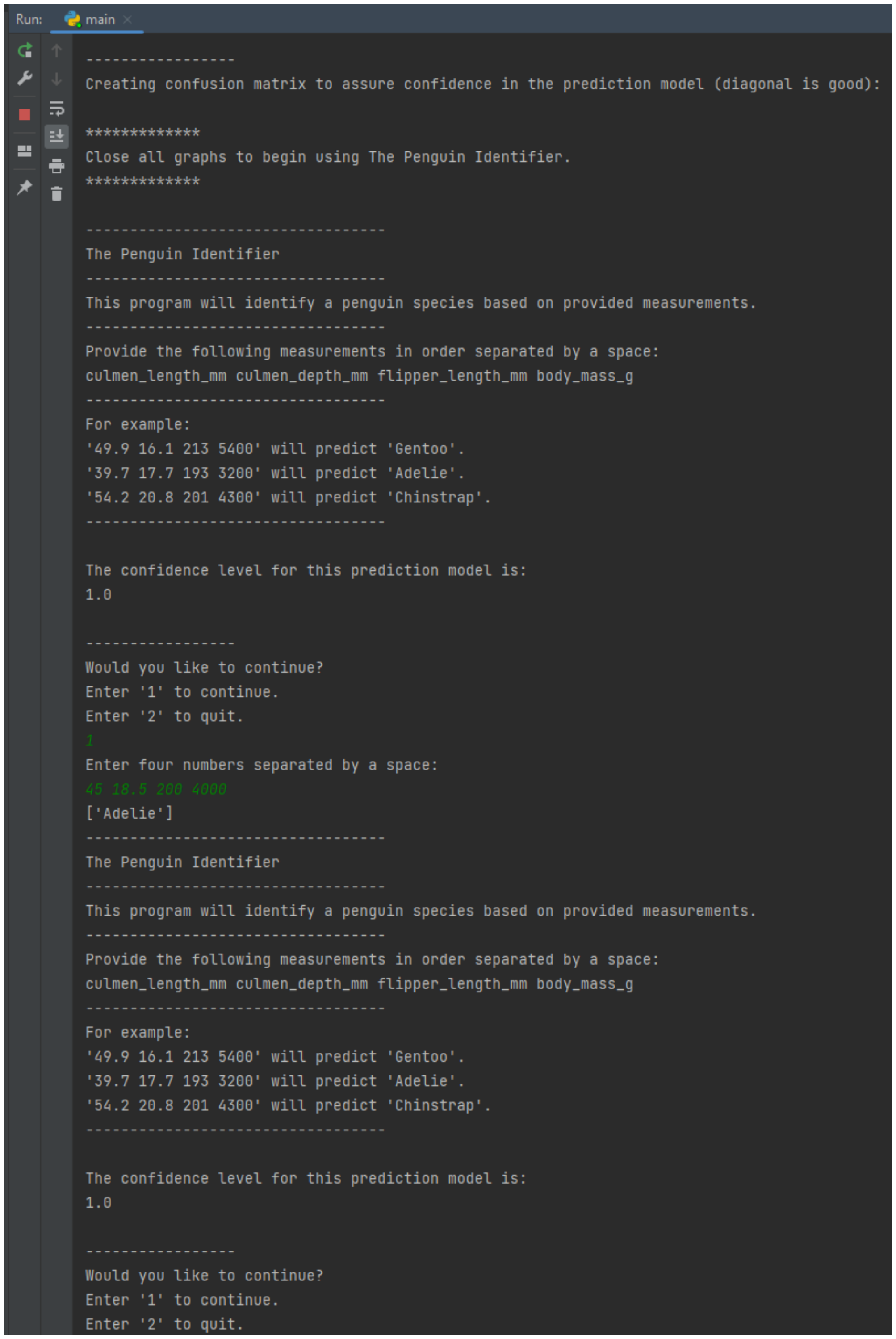
The Penguin Identifier uses a multinominal logistic regression model is pictured below. This model is used to make predictions based on a set of inputs. The model can predict Adelie, Gentoo, or Chinstrap.



```
# NON-DESCRIPTIVE METHOD
# Logistic regression assigns each dependent variable a probability and chooses the variable with the highest
#   probability
penguin_prediction_model = linear_model.LogisticRegression(max_iter=100000)
print("\n----------------")
print("Penguin prediction model has been created using Logistic Regression.")

# Data processing, break into dependent and independent subsets
y_dependent_variable = penguin_data_frame.values[:, 0]
x_independent_variable = penguin_data_frame.values[:, 1:5]
print("\n----------------")
print("Penguin data has been separated into dependent and independent variables. We will be using penguin dimensions "
      "(independent variables) to predict the penguin species (dependent variable).")

# train_test_split returns 4 variables: training and test data for independent & dependent
x_independent_variable_train, x_independent_variable_test, y_dependent_variable_train, y_dependent_variable_test = \
    model_selection.train_test_split(x_independent_variable, y_dependent_variable, test_size=0.2)
print("\n----------------")
print("Training and testing data has been created. 80% of the data will be for training and the remaining 20% will be"
      " for testing.")

# Train model, produces function that maps penguin descriptors (independent) to species (dependent)
# Uses the created training data returned from the train_test_split function
penguin_prediction_model.fit(x_independent_variable_train, y_dependent_variable_train)
print("\n----------------")
print("Penguin prediction model has been fit using the created training data.")
```

- **An interactive "dashboard."**

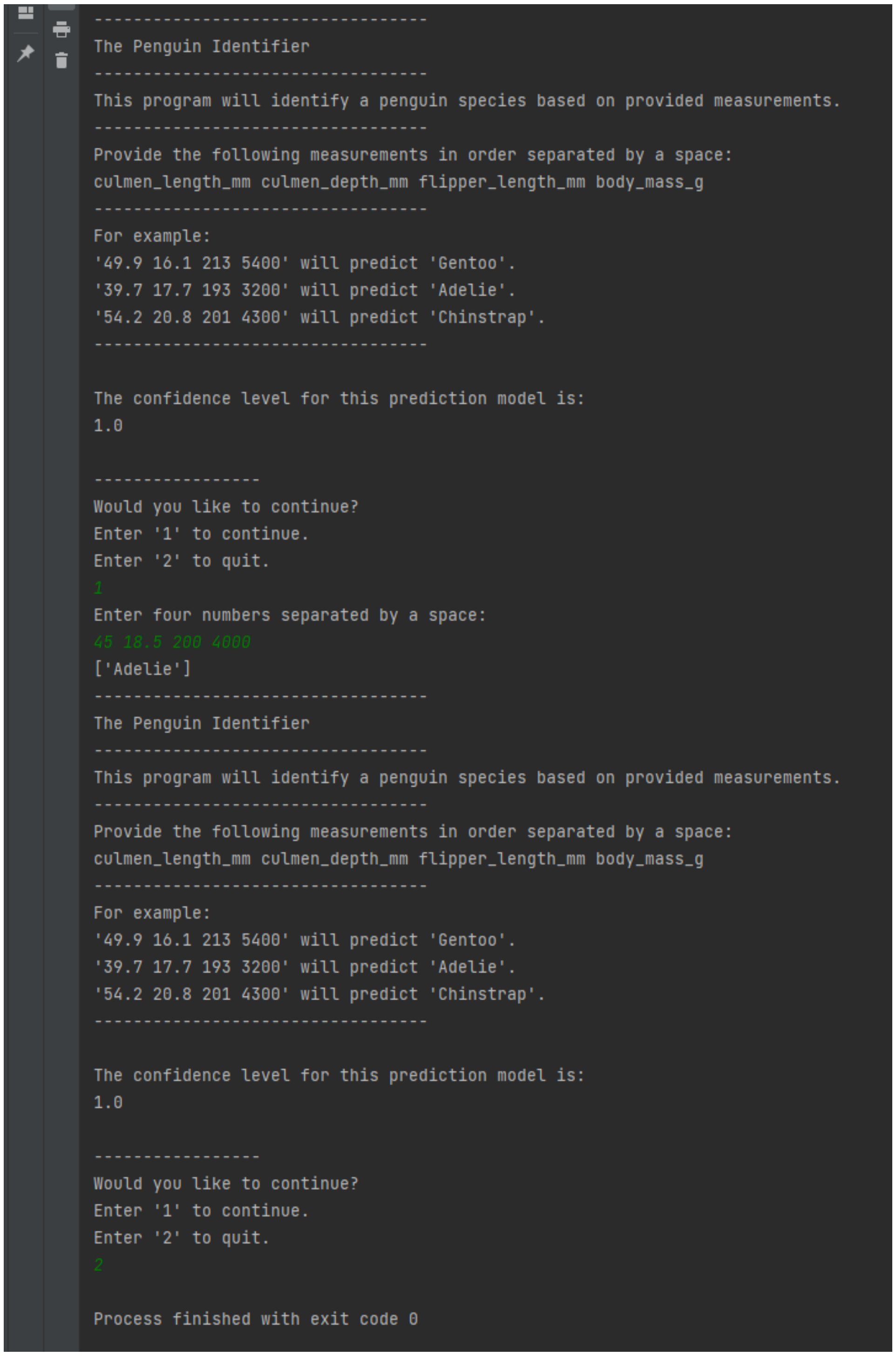
The Penguin Identifier requests input from the user for a penguin's dimensions and weight through the command line interface. The Interaction is pictured below.



```
----------------
Creating confusion matrix to assure confidence in the prediction model (diagonal is good):

*************
Close all graphs to begin using The Penguin Identifier.
*************

---------------------------------
The Penguin Identifier
---------------------------------
This program will identify a penguin species based on provided measurements.
---------------------------------
Provide the following measurements in order separated by a space:
culmen_length_mm culmen_depth_mm flipper_length_mm body_mass_g
---------------------------------
For example:
'49.9 16.1 213 5400' will predict 'Gentoo'.
'39.7 17.7 193 3200' will predict 'Adelie'.
'54.2 20.8 201 4300' will predict 'Chinstrap'.
---------------------------------

The confidence level for this prediction model is:
1.0

----------------
Would you like to continue?
Enter '1' to continue.
Enter '2' to quit.
1
Enter four numbers separated by a space:
45 18.5 200 4000
['Adelie']
---------------------------------
The Penguin Identifier
---------------------------------
This program will identify a penguin species based on provided measurements.
---------------------------------
Provide the following measurements in order separated by a space:
culmen_length_mm culmen_depth_mm flipper_length_mm body_mass_g
---------------------------------
For example:
'49.9 16.1 213 5400' will predict 'Gentoo'.
'39.7 17.7 193 3200' will predict 'Adelie'.
'54.2 20.8 201 4300' will predict 'Chinstrap'.
---------------------------------

The confidence level for this prediction model is:
1.0

----------------
Would you like to continue?
Enter '1' to continue.
Enter '2' to quit.
```

- **A “user-friendly” interface.**

The command line interface gives clear instructions and easy input (1 or 2) to interact with the program. A user guide will also be provided in Part D.

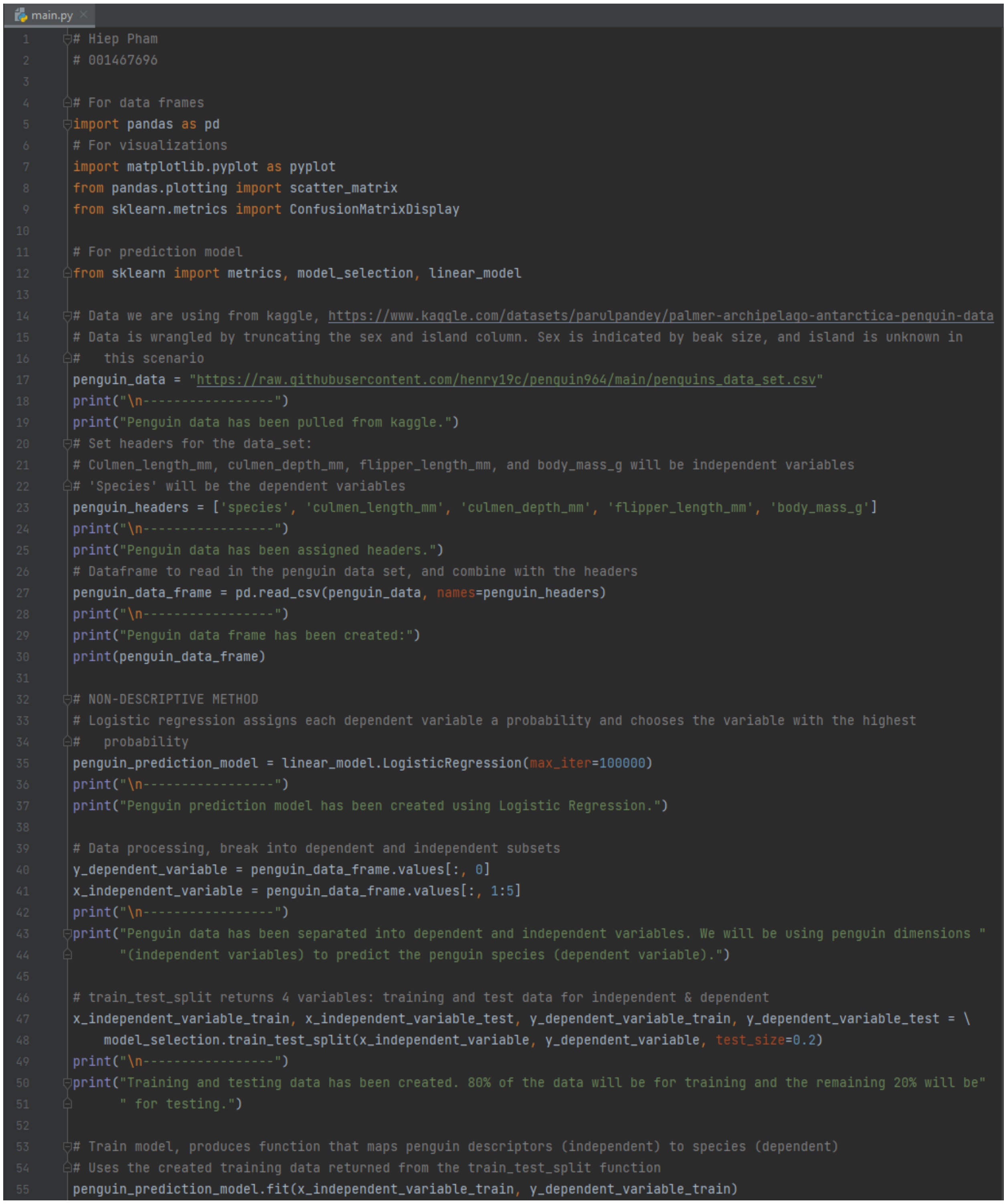


```
----------------------------------
The Penguin Identifier
----------------------------------
This program will identify a penguin species based on provided measurements.
----------------------------------
Provide the following measurements in order separated by a space:
culmen_length_mm culmen_depth_mm flipper_length_mm body_mass_g
----------------------------------
For example:
'49.9 16.1 213 5400' will predict 'Gentoo'.
'39.7 17.7 193 3200' will predict 'Adelie'.
'54.2 20.8 201 4300' will predict 'Chinstrap'.
----------------------------------

The confidence level for this prediction model is:
1.0

-----------------
Would you like to continue?
Enter '1' to continue.
Enter '2' to quit.
1
Enter four numbers separated by a space:
45 18.5 200 4000
['Adelie']
----------------------------------
The Penguin Identifier
----------------------------------
This program will identify a penguin species based on provided measurements.
----------------------------------
Provide the following measurements in order separated by a space:
culmen_length_mm culmen_depth_mm flipper_length_mm body_mass_g
----------------------------------
For example:
'49.9 16.1 213 5400' will predict 'Gentoo'.
'39.7 17.7 193 3200' will predict 'Adelie'.
'54.2 20.8 201 4300' will predict 'Chinstrap'.
----------------------------------

The confidence level for this prediction model is:
1.0

-----------------
Would you like to continue?
Enter '1' to continue.
Enter '2' to quit.
2

Process finished with exit code 0
```

- **Security appropriate to your application's needs.**

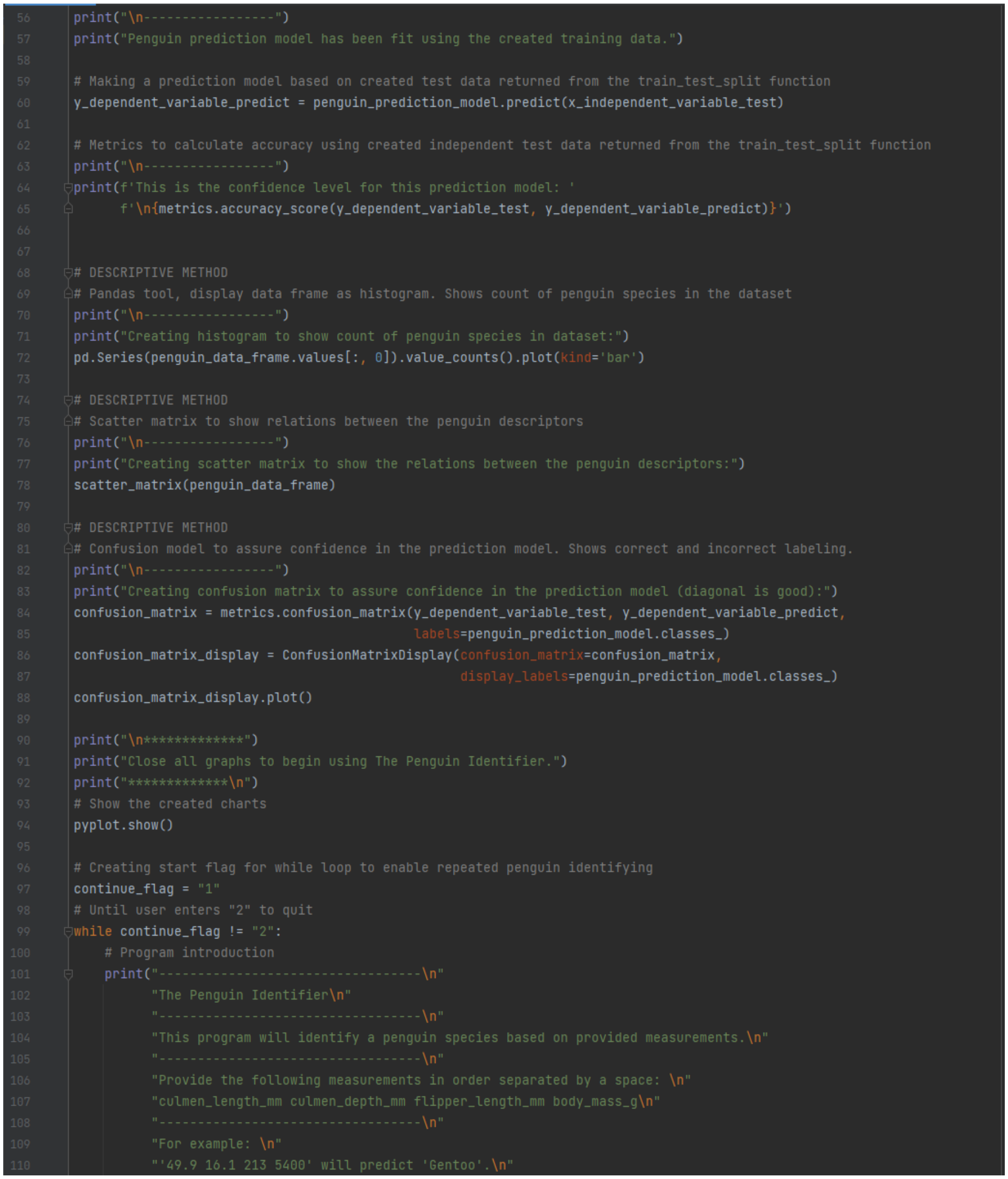
The program is run locally, ensuring minimal attack surface of the application. Because the program does not deal with human data, privacy is not a concern.

The entire code is provided below:

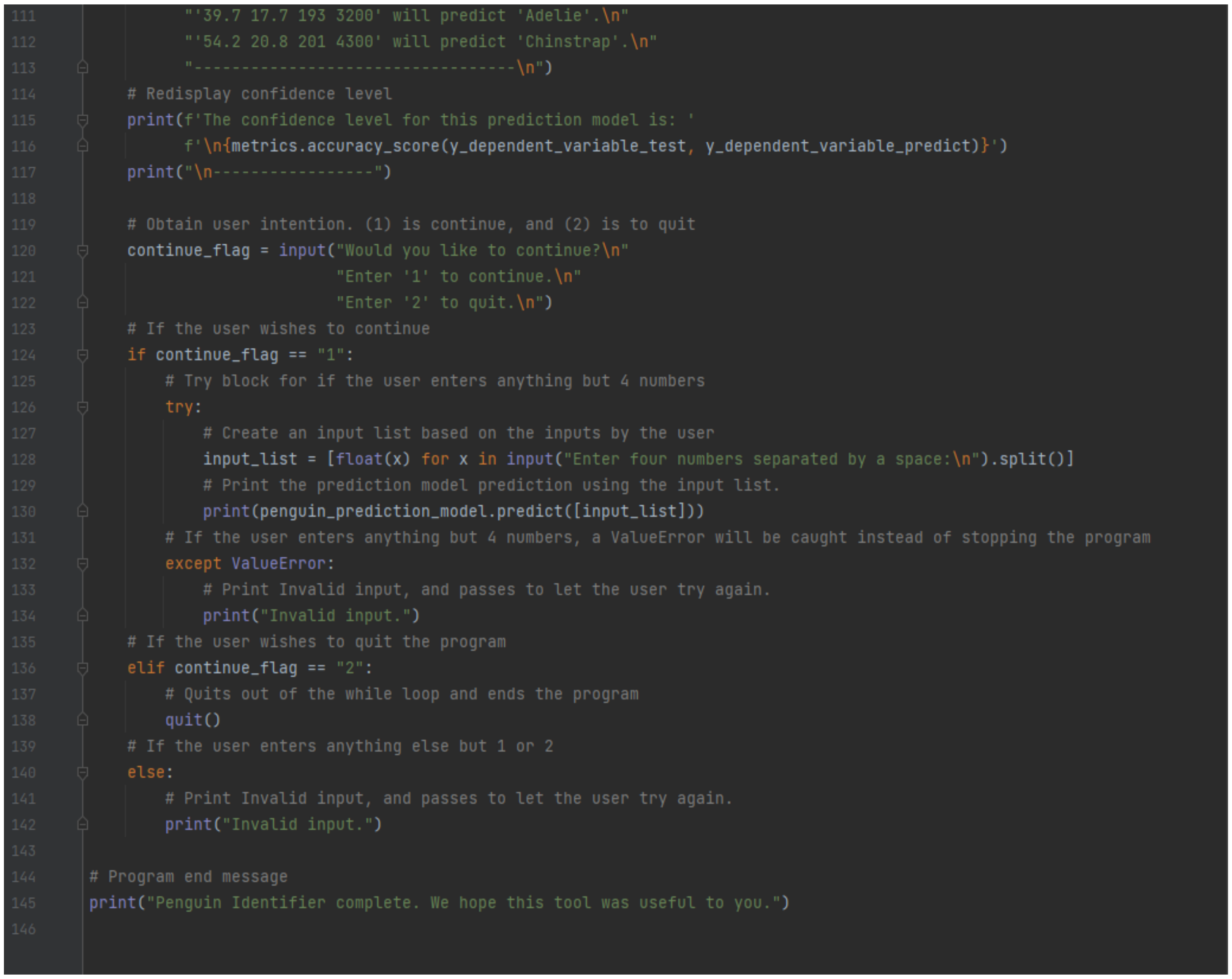


main.py

```python
# Hiep Pham
# 001467696

# For data frames
import pandas as pd
# For visualizations
import matplotlib.pyplot as pyplot
from pandas.plotting import scatter_matrix
from sklearn.metrics import ConfusionMatrixDisplay

# For prediction model
from sklearn import metrics, model_selection, linear_model

# Data we are using from kaggle, https://www.kaggle.com/datasets/parulpandey/palmer-archipelago-antarctica-penguin-data
# Data is wrangled by truncating the sex and island column. Sex is indicated by beak size, and island is unknown in
#   this scenario
penguin_data = "https://raw.githubusercontent.com/henry19c/penguin964/main/penguins_data_set.csv"
print("\n----------------")
print("Penguin data has been pulled from kaggle.")
# Set headers for the data_set:
# Culmen_length_mm, culmen_depth_mm, flipper_length_mm, and body_mass_g will be independent variables
# 'Species' will be the dependent variables
penguin_headers = ['species', 'culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm', 'body_mass_g']
print("\n----------------")
print("Penguin data has been assigned headers.")
# Dataframe to read in the penguin data set, and combine with the headers
penguin_data_frame = pd.read_csv(penguin_data, names=penguin_headers)
print("\n----------------")
print("Penguin data frame has been created:")
print(penguin_data_frame)

# NON-DESCRIPTIVE METHOD
# Logistic regression assigns each dependent variable a probability and chooses the variable with the highest
#   probability
penguin_prediction_model = linear_model.LogisticRegression(max_iter=100000)
print("\n----------------")
print("Penguin prediction model has been created using Logistic Regression.")

# Data processing, break into dependent and independent subsets
y_dependent_variable = penguin_data_frame.values[:, 0]
x_independent_variable = penguin_data_frame.values[:, 1:5]
print("\n----------------")
print("Penguin data has been separated into dependent and independent variables. We will be using penguin dimensions "
      "(independent variables) to predict the penguin species (dependent variable).")

# train_test_split returns 4 variables: training and test data for independent & dependent
x_independent_variable_train, x_independent_variable_test, y_dependent_variable_train, y_dependent_variable_test = \
    model_selection.train_test_split(x_independent_variable, y_dependent_variable, test_size=0.2)
print("\n----------------")
print("Training and testing data has been created. 80% of the data will be for training and the remaining 20% will be"
      " for testing.")

# Train model, produces function that maps penguin descriptors (independent) to species (dependent)
# Uses the created training data returned from the train_test_split function
penguin_prediction_model.fit(x_independent_variable_train, y_dependent_variable_train)
```



```python
print("\n-----------------")
print("Penguin prediction model has been fit using the created training data.")

# Making a prediction model based on created test data returned from the train_test_split function
y_dependent_variable_predict = penguin_prediction_model.predict(x_independent_variable_test)

# Metrics to calculate accuracy using created independent test data returned from the train_test_split function
print("\n-----------------")
print(f'This is the confidence level for this prediction model: '
      f'\n{metrics.accuracy_score(y_dependent_variable_test, y_dependent_variable_predict)}')


# DESCRIPTIVE METHOD
# Pandas tool, display data frame as histogram. Shows count of penguin species in the dataset
print("\n-----------------")
print("Creating histogram to show count of penguin species in dataset:")
pd.Series(penguin_data_frame.values[:, 0]).value_counts().plot(kind='bar')

# DESCRIPTIVE METHOD
# Scatter matrix to show relations between the penguin descriptors
print("\n-----------------")
print("Creating scatter matrix to show the relations between the penguin descriptors:")
scatter_matrix(penguin_data_frame)

# DESCRIPTIVE METHOD
# Confusion model to assure confidence in the prediction model. Shows correct and incorrect labeling.
print("\n-----------------")
print("Creating confusion matrix to assure confidence in the prediction model (diagonal is good):")
confusion_matrix = metrics.confusion_matrix(y_dependent_variable_test, y_dependent_variable_predict,
                                            labels=penguin_prediction_model.classes_)
confusion_matrix_display = ConfusionMatrixDisplay(confusion_matrix=confusion_matrix,
                                                  display_labels=penguin_prediction_model.classes_)
confusion_matrix_display.plot()

print("\n*************")
print("Close all graphs to begin using The Penguin Identifier.")
print("*************\n")
# Show the created charts
pyplot.show()

# Creating start flag for while loop to enable repeated penguin identifying
continue_flag = "1"
# Until user enters "2" to quit
while continue_flag != "2":
    # Program introduction
    print("----------------------------------\n"
          "The Penguin Identifier\n"
          "----------------------------------\n"
          "This program will identify a penguin species based on provided measurements.\n"
          "----------------------------------\n"
          "Provide the following measurements in order separated by a space: \n"
          "culmen_length_mm culmen_depth_mm flipper_length_mm body_mass_g\n"
          "----------------------------------\n"
          "For example: \n"
          "'49.9 16.1 213 5400' will predict 'Gentoo'.\n"
```



```
            "'39.7 17.7 193 3200' will predict 'Adelie'.\n"
            "'54.2 20.8 201 4300' will predict 'Chinstrap'.\n"
            "----------------------------------\n")
    # Redisplay confidence level
    print(f'The confidence level for this prediction model is: '
          f'\n{metrics.accuracy_score(y_dependent_variable_test, y_dependent_variable_predict)}')
    print("\n-----------------")

    # Obtain user intention. (1) is continue, and (2) is to quit
    continue_flag = input("Would you like to continue?\n"
                          "Enter '1' to continue.\n"
                          "Enter '2' to quit.\n")
    # If the user wishes to continue
    if continue_flag == "1":
        # Try block for if the user enters anything but 4 numbers
        try:
            # Create an input list based on the inputs by the user
            input_list = [float(x) for x in input("Enter four numbers separated by a space:\n").split()]
            # Print the prediction model prediction using the input list.
            print(penguin_prediction_model.predict([input_list]))
        # If the user enters anything but 4 numbers, a ValueError will be caught instead of stopping the program
        except ValueError:
            # Print Invalid input, and passes to let the user try again.
            print("Invalid input.")
    # If the user wishes to quit the program
    elif continue_flag == "2":
        # Quits out of the while loop and ends the program
        quit()
    # If the user enters anything else but 1 or 2
    else:
        # Print Invalid input, and passes to let the user try again.
        print("Invalid input.")

# Program end message
print("Penguin Identifier complete. We hope this tool was useful to you.")
```

Part D: Post-implementation Report

A Business (or Organization) Vision

The Denver Zoo has received a mass number of penguins from the melted Antarctica polar cap. The Zoo has received the Adelie, Gentoo, and Chinstrap penguins. Unfortunately, due to the sheer amount of colonies, the penguins have intermingled and mixed. This can potentially cause territory and colonial wars between the penguins, resulting in danger for the zookeepers and penguins alike. The Denver Zoo would greatly benefit from a solution that enables efficient identification of penguin species based on penguin measurements.

The Penguin Identifier solves this exact issue. The Penguin Identifier identifies a penguin species based on culmen length, culmen depth, flipper length, and body mass.

A user can input these measurements for any Adelie, Gentoo, or Chinstrap penguin and the program will use a Logistic Regression prediction model to predict the penguin species based on the input. The user can then confidently identify/tag the penguin with the correct species and separate the penguin accordingly.

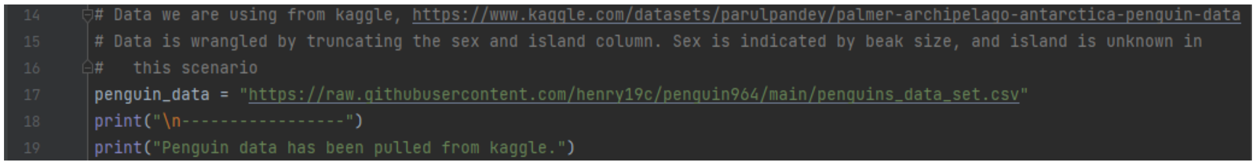
As an example, zookeeper Kelly oversees separating the penguins based on their species to maintain their colonies. Kelly requests zookeeper Bob to begin collecting the penguins in a line and measuring them. As Kelly receives the penguins from Bob, she inputs the measurements provided by Bob into the Penguin Identifier. Kelly enters “1” to continue identifying a penguin and proceeds to enter “49.9 16.1 213 5400”. Immediately, the program responds “Gentoo”. With the confidence of 90%+, as displayed by the application, Kelly can feel assured the penguins are categorized correctly and proceed to pass the penguin to zookeeper Terry to lead the penguin back to its appropriate colony.

Datasets

The penguin dataset used to train the Logistic Regression prediction model was sourced from "Palmer Archipelago (Antarctica) penguin data", created by Parul Pandey on www.kaggle.com (Gorman KB). From this dataset, we specifically use the "penguins_size.csv" CSV file. This file contains a given penguin's species, culmen_length_mm, culmen_depth_mm, body_mass_g, island, and sex.

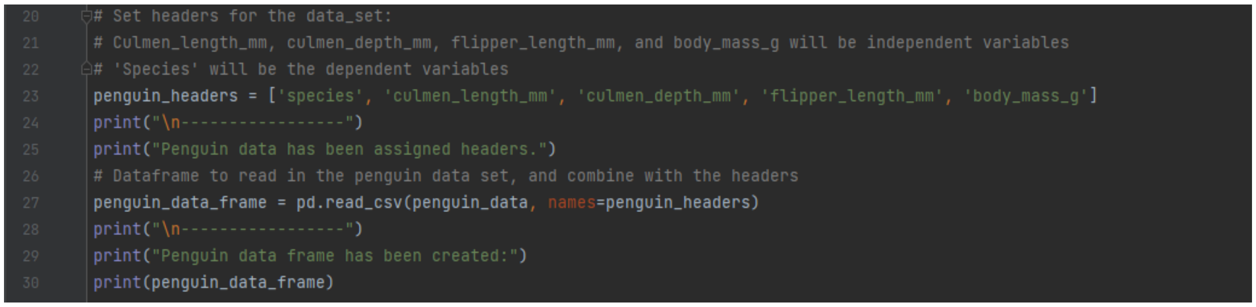
The data is processed by truncating 2 columns and creating "pandas" data frames from the dataset. In the article "To tell the sex of a Galápagos penguin, measure its beak, researchers say" published by the University of Washington, author James Urton says, "beak size is nearly a perfect indicator of whether a bird is male or female" (Urton). Because of this, we choose to omit the sex column from the dataset, because we are measuring the culmen (upper ridge of the penguin's beak). This eliminates 1 step for the zookeeper to classify the penguin's species using this program. Additionally, we also omit the "island" column from the dataset. Due to the circumstances, mass amounts of penguins are imported from all over, and it was infeasible to track from which island each bird originated. Hence, the island is omitted when inputting data for a species prediction.

Once we have omitted those two columns, we also omit the headers labeling the columns. The exclusion of the labels makes data processing simpler when writing the Python application. We then upload the remaining data (species, culmen length, culmen depth, flipper length, and body mass) onto a GitHub repository "penguins964". On the Python IDE (in this case, we're using PyCharm), we "pull" the data by reading in the online CSV using the following code:



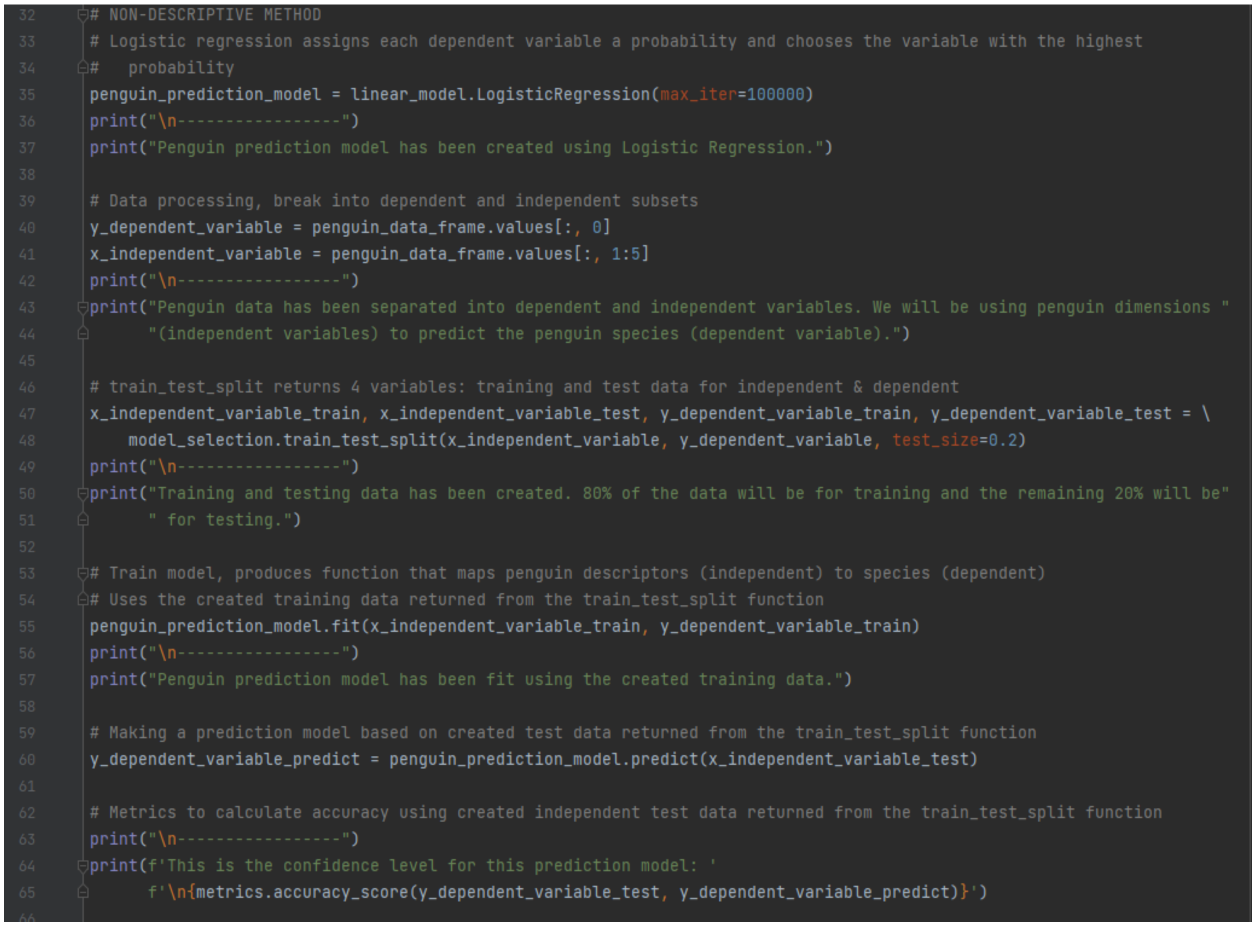
```
# Data we are using from kaggle, https://www.kaggle.com/datasets/parulpandey/palmer-archipelago-antarctica-penguin-data
# Data is wrangled by truncating the sex and island column. Sex is indicated by beak size, and island is unknown in
#   this scenario
penguin_data = "https://raw.githubusercontent.com/henry19c/penguin964/main/penguins_data_set.csv"
print("\n----------------")
print("Penguin data has been pulled from kaggle.")
```

We then recreate the headers for the “table”, and create our “pandas” data frame from the data we pulled and the header we created:



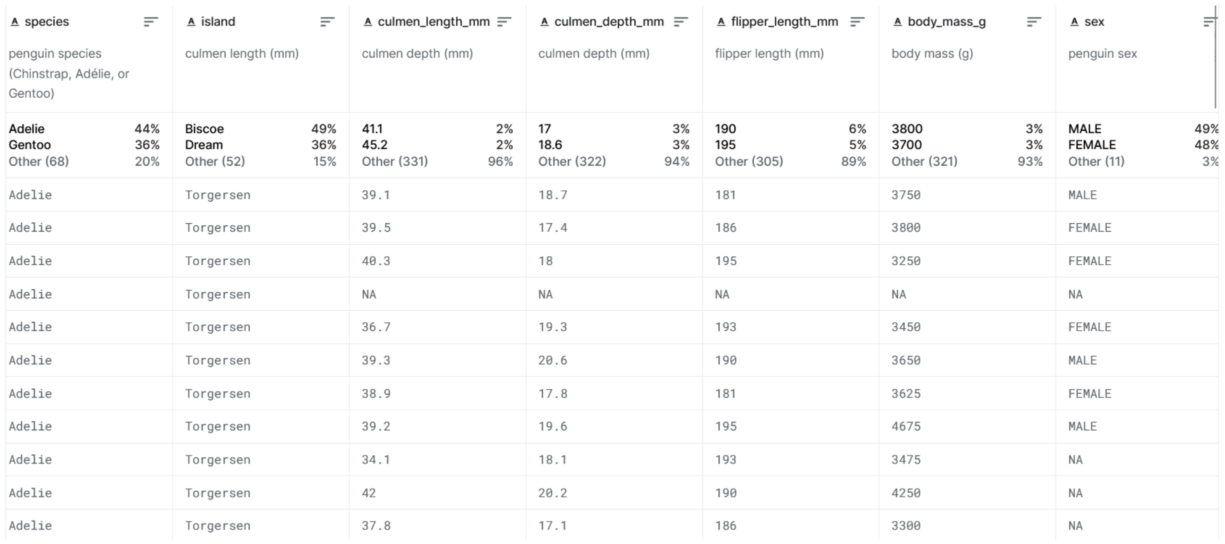
```python
# Set headers for the data_set:
# Culmen_length_mm, culmen_depth_mm, flipper_length_mm, and body_mass_g will be independent variables
# 'Species' will be the dependent variables
penguin_headers = ['species', 'culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm', 'body_mass_g']
print("\n-----------------")
print("Penguin data has been assigned headers.")
# Dataframe to read in the penguin data set, and combine with the headers
penguin_data_frame = pd.read_csv(penguin_data, names=penguin_headers)
print("\n-----------------")
print("Penguin data frame has been created:")
print(penguin_data_frame)
```

This data frame will be used to create our prediction model, as well as training and testing data to provide a confidence level to the prediction model. A y_dependent_variable is created to contain all the penguin species, and an x_independent_variable to contain the penguin attributes.



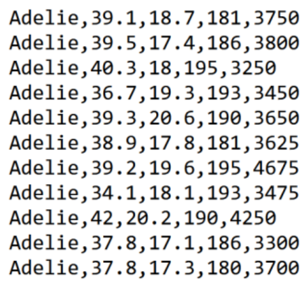
```python
# NON-DESCRIPTIVE METHOD
# Logistic regression assigns each dependent variable a probability and chooses the variable with the highest
#   probability
penguin_prediction_model = linear_model.LogisticRegression(max_iter=100000)
print("\n-----------------")
print("Penguin prediction model has been created using Logistic Regression.")

# Data processing, break into dependent and independent subsets
y_dependent_variable = penguin_data_frame.values[:, 0]
x_independent_variable = penguin_data_frame.values[:, 1:5]
print("\n-----------------")
print("Penguin data has been separated into dependent and independent variables. We will be using penguin dimensions "
      "(independent variables) to predict the penguin species (dependent variable).")

# train_test_split returns 4 variables: training and test data for independent & dependent
x_independent_variable_train, x_independent_variable_test, y_dependent_variable_train, y_dependent_variable_test = \
    model_selection.train_test_split(x_independent_variable, y_dependent_variable, test_size=0.2)
print("\n-----------------")
print("Training and testing data has been created. 80% of the data will be for training and the remaining 20% will be"
      " for testing.")

# Train model, produces function that maps penguin descriptors (independent) to species (dependent)
# Uses the created training data returned from the train_test_split function
penguin_prediction_model.fit(x_independent_variable_train, y_dependent_variable_train)
print("\n-----------------")
print("Penguin prediction model has been fit using the created training data.")

# Making a prediction model based on created test data returned from the train_test_split function
y_dependent_variable_predict = penguin_prediction_model.predict(x_independent_variable_test)

# Metrics to calculate accuracy using created independent test data returned from the train_test_split function
print("\n-----------------")
print(f'This is the confidence level for this prediction model: '
      f'\n{metrics.accuracy_score(y_dependent_variable_test, y_dependent_variable_predict)}')
```

The following is an example of the raw data, shown directly on the www.kaggle.com site:



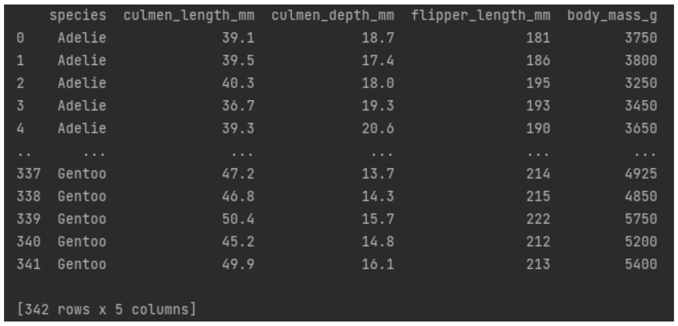
species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
penguin species (Chinstrap, Adélie, or Gentoo)	culmen length (mm)	culmen depth (mm)	culmen depth (mm)	flipper length (mm)	body mass (g)	penguin sex
Adelie 44%	Biscoe 49%	41.1 2%	17 3%	190 6%	3800 3%	MALE 49%
Gentoo 36%	Dream 36%	45.2 2%	18.6 3%	195 5%	3700 3%	FEMALE 48%
Other (68) 20%	Other (52) 15%	Other (331) 96%	Other (322) 94%	Other (305) 89%	Other (321) 93%	Other (11) 3%
Adelie	Torgersen	39.1	18.7	181	3750	MALE
Adelie	Torgersen	39.5	17.4	186	3800	FEMALE
Adelie	Torgersen	40.3	18	195	3250	FEMALE
Adelie	Torgersen	NA	NA	NA	NA	NA
Adelie	Torgersen	36.7	19.3	193	3450	FEMALE
Adelie	Torgersen	39.3	20.6	190	3650	MALE
Adelie	Torgersen	38.9	17.8	181	3625	FEMALE
Adelie	Torgersen	39.2	19.6	195	4675	MALE
Adelie	Torgersen	34.1	18.1	193	3475	NA
Adelie	Torgersen	42	20.2	190	4250	NA
Adelie	Torgersen	37.8	17.1	186	3300	NA

We then truncate the island and sex columns (reasons found above), and eliminated the header row:



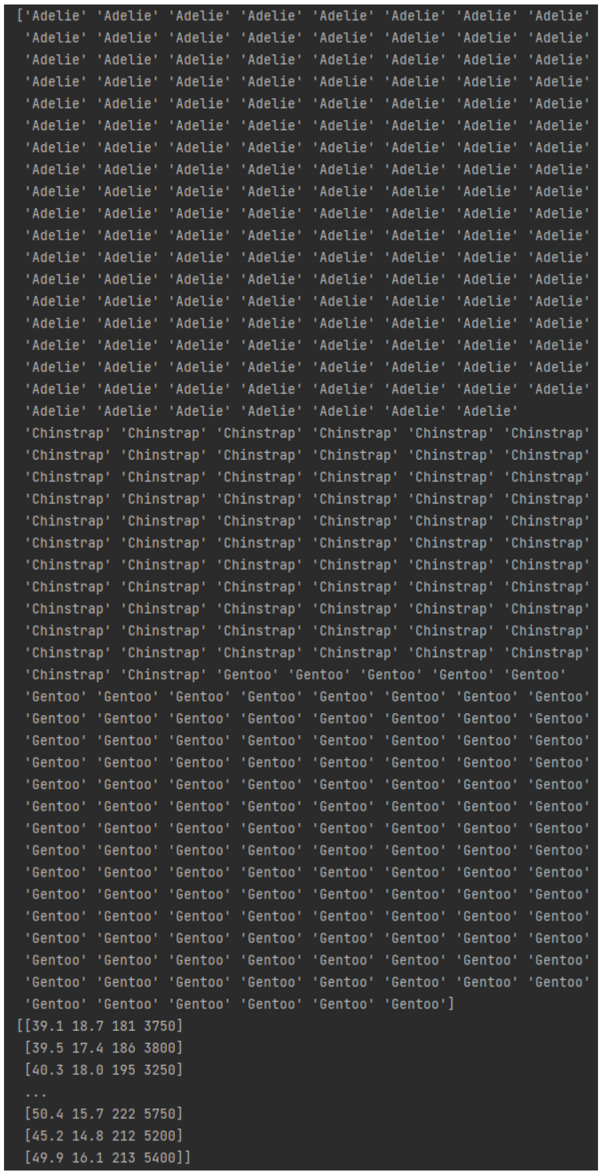
```
Adelie,39.1,18.7,181,3750
Adelie,39.5,17.4,186,3800
Adelie,40.3,18,195,3250
Adelie,36.7,19.3,193,3450
Adelie,39.3,20.6,190,3650
Adelie,38.9,17.8,181,3625
Adelie,39.2,19.6,195,4675
Adelie,34.1,18.1,193,3475
Adelie,42,20.2,190,4250
Adelie,37.8,17.1,186,3300
Adelie,37.8,17.3,180,3700
```

Then, we process this data and turned them into data frames. The following is the penguin_data_frame, with the headers reattached:



```
    species  culmen_length_mm  culmen_depth_mm  flipper_length_mm  body_mass_g
0    Adelie              39.1             18.7                181         3750
1    Adelie              39.5             17.4                186         3800
2    Adelie              40.3             18.0                195         3250
3    Adelie              36.7             19.3                193         3450
4    Adelie              39.3             20.6                190         3650
..      ...               ...              ...                ...          ...
337  Gentoo              47.2             13.7                214         4925
338  Gentoo              46.8             14.3                215         4850
339  Gentoo              50.4             15.7                222         5750
340  Gentoo              45.2             14.8                212         5200
341  Gentoo              49.9             16.1                213         5400

[342 rows x 5 columns]
```

Lastly, we separate this data into 2 additional data frames, the y_dependent_variable and x_independent variable:

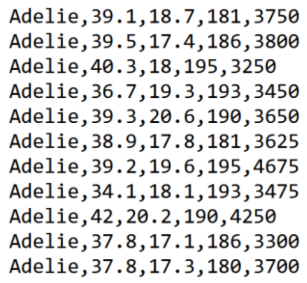


```
['Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap'
 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap'
 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap'
 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap'
 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap'
 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap'
 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap'
 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap'
 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap'
 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap'
 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap'
 'Chinstrap' 'Chinstrap' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo']
[[39.1 18.7 181 3750]
 [39.5 17.4 186 3800]
 [40.3 18.0 195 3250]
 ...
 [50.4 15.7 222 5750]
 [45.2 14.8 212 5200]
 [49.9 16.1 213 5400]]
```

The raw dataset can be accessed via "https://www.kaggle.com/datasets/parulpandey/palmer-archipelago-antarctica-penguin-data?select=penguins_size.csv", and the processed data can be accessed on GitHub via "https://raw.githubusercontent.com/henry19c/penguin964/main/penguins_data_set.csv".

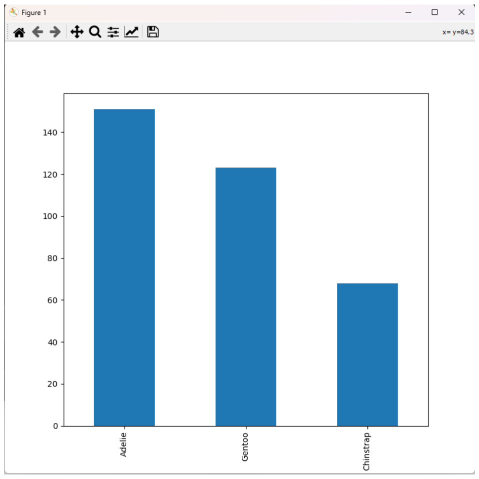
Data Product Code

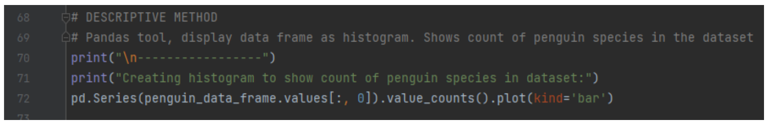
The raw data was sourced from “Palmer Archipelago (Antarctica) penguin data”, created by Parul Pandey on www.kaggle.com (Gorman KB). This raw data is processed by truncating columns that were not relevant to the scenario. One such omission is the column “island”. Due to the scenario, the mass influx of penguins could come from all over the place, and it was logistically infeasible to mark all the penguins’ island origination. After all, the health and safety of the penguins were the #1 priority during the melting of the polar ice caps. The other omission is the column “sex” because in the article “To tell the sex of a Galápagos penguin, measure its beak, researchers say” published by the University of Washington, author James Urton says, “beak size is nearly a perfect indicator of whether a bird is male or female” (Urton). Because we are collecting the culmen sizes, we do not need to duplicate efforts just to determine the sex. Furthermore, and perhaps more importantly, we are simply segregating species, not sex. The processed data is shown below:



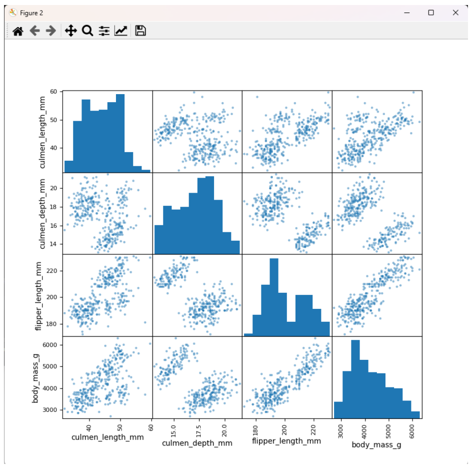
```
Adelie,39.1,18.7,181,3750
Adelie,39.5,17.4,186,3800
Adelie,40.3,18,195,3250
Adelie,36.7,19.3,193,3450
Adelie,39.3,20.6,190,3650
Adelie,38.9,17.8,181,3625
Adelie,39.2,19.6,195,4675
Adelie,34.1,18.1,193,3475
Adelie,42,20.2,190,4250
Adelie,37.8,17.1,186,3300
Adelie,37.8,17.3,180,3700
```

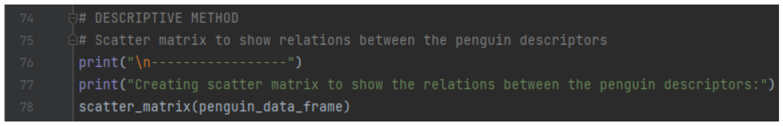
Two descriptive methods describe the data we are processing/using. These descriptive methods also create corresponding visualizations. The first of which is a histogram showing the count of penguin species in the dataset. This histogram will ensure the user that the dataset we are using consider all three penguin species and will display if a certain species is not accounted for enough. The second descriptive method is the creation of a scatter matrix. The scatter matrix shows the relationship between different penguin measurements. The visualization demonstrates visual representations of culmen length, culmen depth, flipper length, and body mass correlations. As an example, a user can observe the scatter matrix and make the determination that as flipper length grows, body mass does too (which would make clear sense). Secondly, for the developer, the scatter matrix ensures that only relevant data is processed while developing the model. If there is a clear missing correlation, the developer could reconsider whether the data was truly relevant to the model. The developer would then need to discuss this with the team and client, hopefully during the Requirements or Design phase. Both descriptive methods and visualizations are shown below:





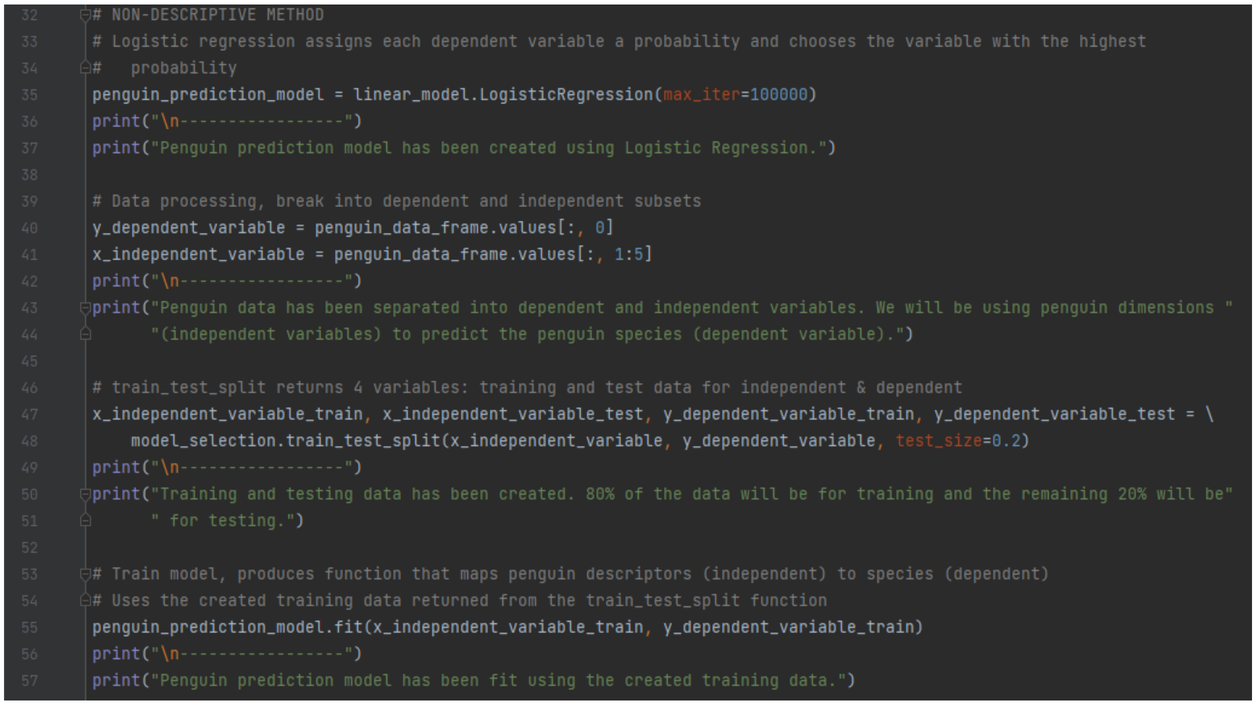
```
# DESCRIPTIVE METHOD
# Pandas tool, display data frame as histogram. Shows count of penguin species in the dataset
print("\n----------------")
print("Creating histogram to show count of penguin species in dataset:")
pd.Series(penguin_data_frame.values[:, 0]).value_counts().plot(kind='bar')
```





```
# DESCRIPTIVE METHOD
# Scatter matrix to show relations between the penguin descriptors
print("\n----------------")
print("Creating scatter matrix to show the relations between the penguin descriptors:")
scatter_matrix(penguin_data_frame)
```

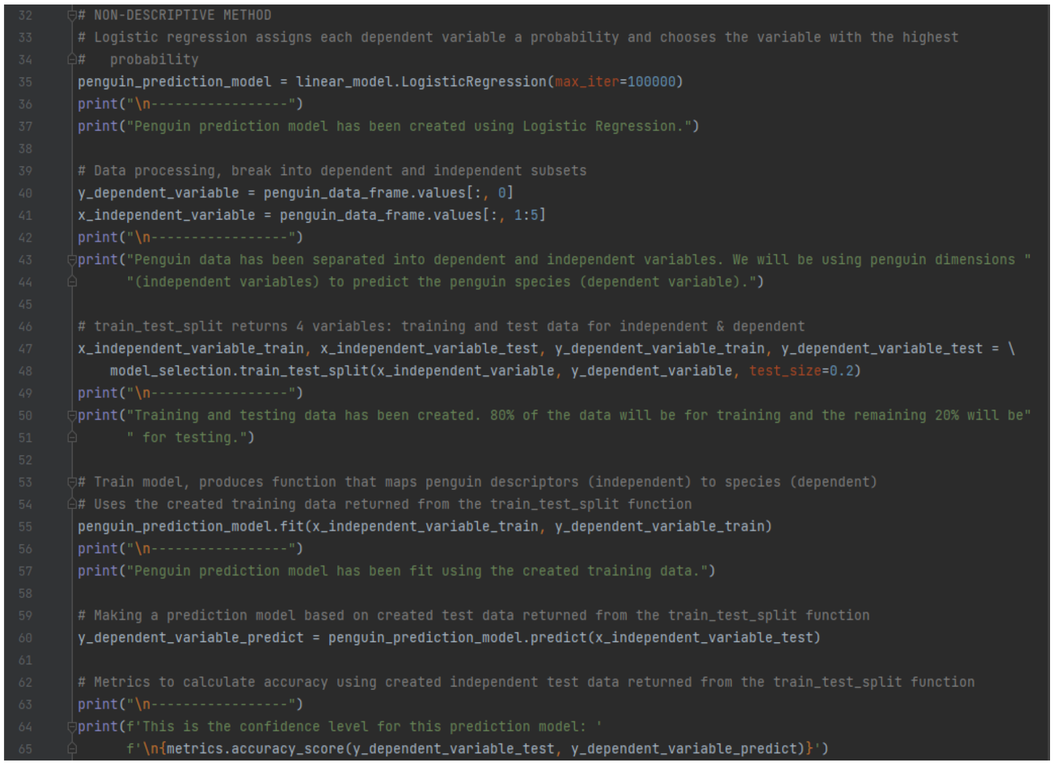
The non-descriptive method utilized in the Penguin Identifier is the Logistic Regression model. The method does not describe the dataset; in fact, it utilizes the independent data of the dataset to create a new prediction of the dependent value of the dataset. In other words, the Logistic Regression model assigns all three species with a probability of being the “correct” species based on the user’s inputted penguin dimensions. Then, the prediction model outputs the category with the greatest probability as its prediction. The Logistic Regression is derived from the “scikit-learn" module and uses independent variables to determine the dependent variable. The code is shown below:



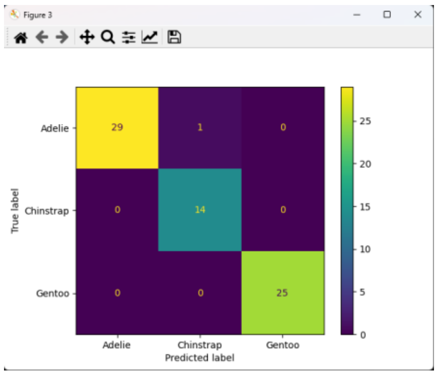
```python
# NON-DESCRIPTIVE METHOD
# Logistic regression assigns each dependent variable a probability and chooses the variable with the highest
#   probability
penguin_prediction_model = linear_model.LogisticRegression(max_iter=100000)
print("\n----------------")
print("Penguin prediction model has been created using Logistic Regression.")

# Data processing, break into dependent and independent subsets
y_dependent_variable = penguin_data_frame.values[:, 0]
x_independent_variable = penguin_data_frame.values[:, 1:5]
print("\n----------------")
print("Penguin data has been separated into dependent and independent variables. We will be using penguin dimensions "
      "(independent variables) to predict the penguin species (dependent variable).")

# train_test_split returns 4 variables: training and test data for independent & dependent
x_independent_variable_train, x_independent_variable_test, y_dependent_variable_train, y_dependent_variable_test = \
    model_selection.train_test_split(x_independent_variable, y_dependent_variable, test_size=0.2)
print("\n----------------")
print("Training and testing data has been created. 80% of the data will be for training and the remaining 20% will be"
      " for testing.")

# Train model, produces function that maps penguin descriptors (independent) to species (dependent)
# Uses the created training data returned from the train_test_split function
penguin_prediction_model.fit(x_independent_variable_train, y_dependent_variable_train)
print("\n----------------")
print("Penguin prediction model has been fit using the created training data.")
```

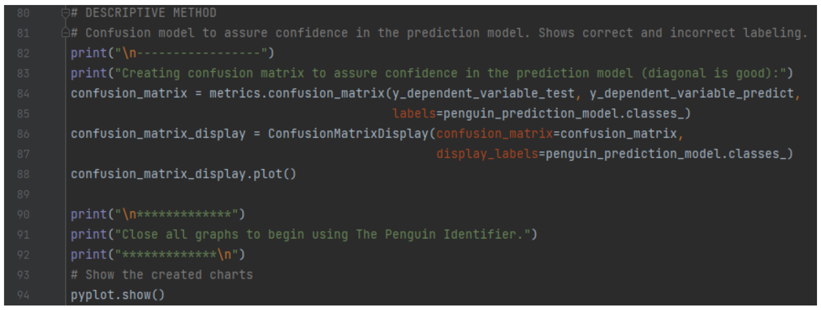
The analytic method applied in the Penguin Identifier application is Logistic Regression. Logistic Regression is a linear classification algorithm meant to predict a dependent variable based on independent variables. Logistic Regressions excel in multiclass classification scenarios, such as this penguin identification situation. Secondly, another reason for the choice of Logistic Regression is its implementation simplicity and efficiency. Given the urgent nature of the situation, quick and efficient solutions are prioritized for seamless deployment. Hence, the Logistic Regression is perfect for the scenario because the Denver Zoo has received a mass influx of penguins and needs to categorize them based on their species. The Denver Zoo has penguins themselves, and the dimensions can be measured by the zookeepers during in-processing. This information will be utilized as the independent variables to predict the penguin species (dependent variables).

The Logistic Regression prediction model was trained using “scikit-learn’s” native train_test_split and fit tool. The train_test_split tool separates the dataset into two portions: one meant for training the prediction model, and the other for testing the prediction model with known correct data. In the Penguin Identifier, 80% of the dataset will be reserved for training the model, while the remaining 20% will be reserved for testing the model. The train_test_split function will return 4 variables: the independent training data, the dependent training data, the independent test data, and the dependent test data. Once all four variables are created, we will fit() the prediction model using the independent and dependent training data. To test the data, we also use “scikit-learn’s” native tool metrics.accuracy_score(). We input the dependent test data we created from train_test_split and test to see how well it does against the prediction model populated with the independent test data. metrics.accuracy_score() will output a value from 0.0-1.0, representing the decimal value of the percentage of correct predictions. In our testing, we have yet to see it drop under .90, as required in our Requirements phase. Furthermore, we also create the Confusion Matrix to visualize the accuracy of the model’s prediction. The confusion matrix displays all the correctly predicted penguins (on the diagonal) and the incorrectly predicted penguins (elsewhere). The creation and testing of the Logistic Regression code are shown below, as well as the confusion matrix and its code:



```python
# NON-DESCRIPTIVE METHOD
# Logistic regression assigns each dependent variable a probability and chooses the variable with the highest
#   probability
penguin_prediction_model = linear_model.LogisticRegression(max_iter=100000)
print("\n-----------------")
print("Penguin prediction model has been created using Logistic Regression.")

# Data processing, break into dependent and independent subsets
y_dependent_variable = penguin_data_frame.values[:, 0]
x_independent_variable = penguin_data_frame.values[:, 1:5]
print("\n-----------------")
print("Penguin data has been separated into dependent and independent variables. We will be using penguin dimensions "
      "(independent variables) to predict the penguin species (dependent variable).")

# train_test_split returns 4 variables: training and test data for independent & dependent
x_independent_variable_train, x_independent_variable_test, y_dependent_variable_train, y_dependent_variable_test = \
    model_selection.train_test_split(x_independent_variable, y_dependent_variable, test_size=0.2)
print("\n-----------------")
print("Training and testing data has been created. 80% of the data will be for training and the remaining 20% will be"
      " for testing.")

# Train model, produces function that maps penguin descriptors (independent) to species (dependent)
# Uses the created training data returned from the train_test_split function
penguin_prediction_model.fit(x_independent_variable_train, y_dependent_variable_train)
print("\n-----------------")
print("Penguin prediction model has been fit using the created training data.")

# Making a prediction model based on created test data returned from the train_test_split function
y_dependent_variable_predict = penguin_prediction_model.predict(x_independent_variable_test)

# Metrics to calculate accuracy using created independent test data returned from the train_test_split function
print("\n-----------------")
print(f'This is the confidence level for this prediction model: '
      f'\n{metrics.accuracy_score(y_dependent_variable_test, y_dependent_variable_predict)}')
```





```
# DESCRIPTIVE METHOD
# Confusion model to assure confidence in the prediction model. Shows correct and incorrect labeling.
print("\n----------------")
print("Creating confusion matrix to assure confidence in the prediction model (diagonal is good):")
confusion_matrix = metrics.confusion_matrix(y_dependent_variable_test, y_dependent_variable_predict,
                                            labels=penguin_prediction_model.classes_)
confusion_matrix_display = ConfusionMatrixDisplay(confusion_matrix=confusion_matrix,
                                                  display_labels=penguin_prediction_model.classes_)
confusion_matrix_display.plot()

print("\n*************")
print("Close all graphs to begin using The Penguin Identifier.")
print("*************\n")
# Show the created charts
pyplot.show()
```

Data analysis was performed through descriptive methods such as the histogram and scatter matrix. The histogram displayed the count of penguin species and verified that all relevant penguin species were accounted for. The histogram ensured developers and zoologists can verify the dataset's adequacy in representing all relevant penguin colonies that were imported to the zoo. The histogram would've displayed if any species were underrepresented and raised concerns to the development team that the prediction model could be skewed/inaccurate. The histogram reassures the team that the dataset was well-balanced and could be trusted to train the prediction model. Meanwhile, the scatter matrix allowed the development team and zoologists to observe the relationship between different penguin dimensions (culmen length, culmen depth, flipper length, and body mass). The scatter matrix plots these measurements against each other to show the correlation between each measurement and enabled the user to determine correlations such as flipper length growing alongside body mass (which makes sense). As an interesting note, 6 of the 12 scatter graphs show 2 separate correlation lines, which likely indicate differences in species development, such as one species having genetically smaller culmen depth compared to another species despite being at the same body mass.

Data analysis was also performed during the creation of our non-descriptive method. Notably, the exclusion of the "island" and "sex" columns was due to analyzing the data and making the determination that these columns were not relevant to the scenario. It was infeasible to track the penguins' island origin, and determining the sex would've duplicated the efforts of determining the species. We also analyze the data when testing for the accuracy of the prediction model using tools such as train_test_split() and metrics.accuracy_score(). These functions create training and test data and use this data to determine the accuracy of the developed prediction model. This enables our development team to course-correct if the accuracy is consistently falling below agreed-upon standards.

The raw dataset can be accessed via "https://www.kaggle.com/datasets/parulpandey/palmer-archipelago-antarctica-penguin-data?select=penguins_size.csv", and the processed data can be accessed on GitHub via "https://raw.githubusercontent.com/henry19c/penguin964/main/penguins_data_set.csv". The project code itself is also attached via direct submission.

Objective (or Hypothesis) Verification

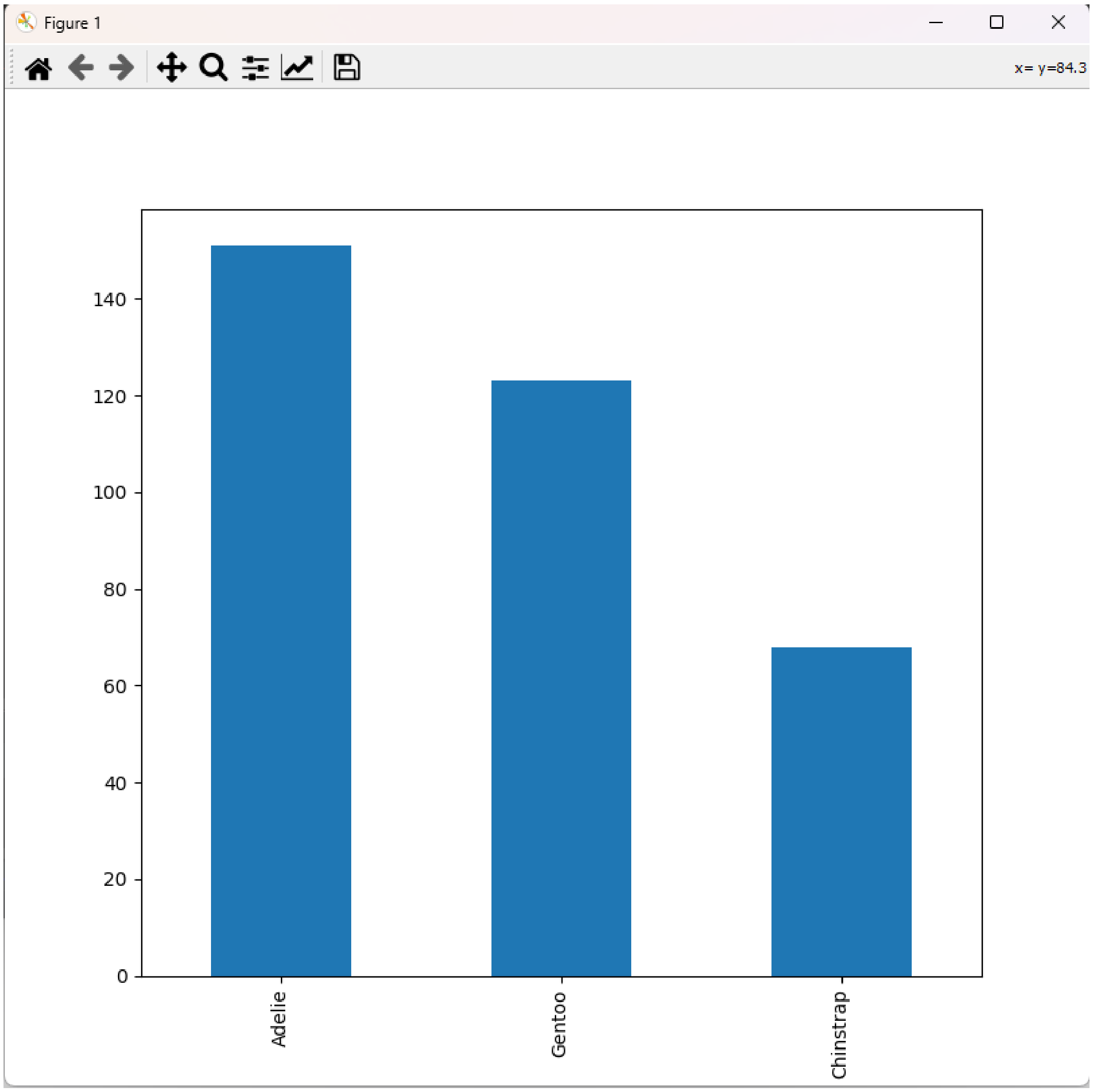
The objective of the Penguin Identifier program was to provide the Denver Zoo with a reliable solution to identify and segregate the mass import of Adelie, Gentoo, and Chinstrap penguins based on their measurements. The desired outcome of implementing the Penguin Identifier was the avoidance of territorial and colonial conflicts between the intertwined penguin species, maintaining the safety and health of the penguins and zookeepers alike.

The hypothesis was: Provided a reliable dataset, the Penguin Identifier, utilizing a Logistic Regression model, will accurately predict penguin species with a confidence level of 90% or greater. We tested our hypothesis using "scikit-learn's" metrics.accuracy_score tool. Using this tool, we tested our program hundreds of times during development, and not once did we see the confidence level drop below 90%. Because of this, we can confidently claim the hypothesis was met.

Effective Visualization and Reporting

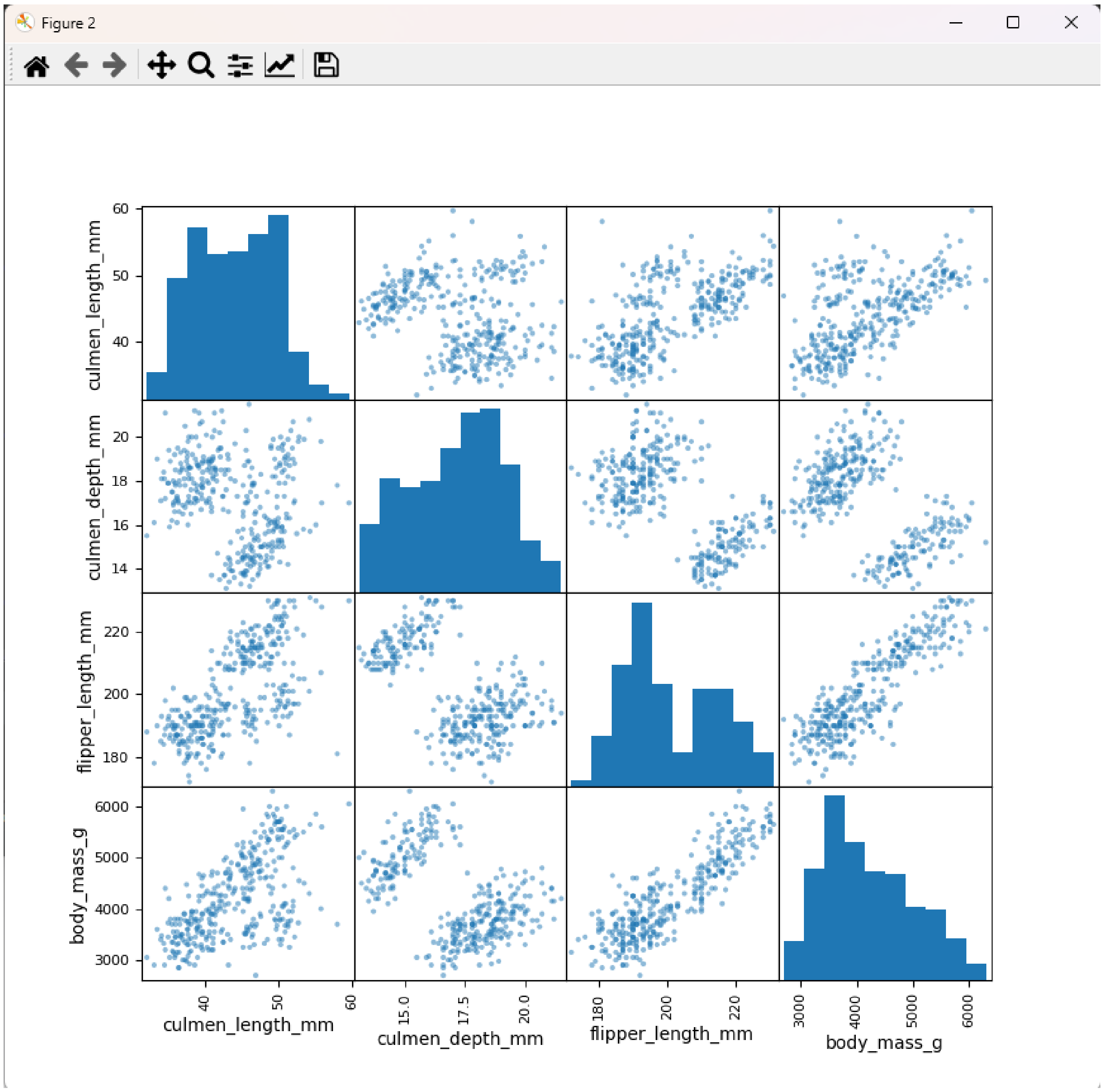
Initially, the dataset is parsed manually by our team of developers for any missing data. We noted that there were some data excluded in the “sex” column, though this is not relevant to our scenario since we are not using “sex” as an independent variable for our prediction model.

After parsing the dataset manually and confirming the data was complete and relevant, we graphed the count of each species using a histogram. This histogram ensured the user that the dataset we are using considers all three penguin species and will display if a certain species is not accounted for enough. The histogram displayed the count of penguin species and verified that all relevant penguin species were accounted for. The histogram ensured developers and zoologists can verify the dataset’s adequacy in representing all relevant penguin colonies that were imported to the zoo. The histogram would’ve displayed if any species were underrepresented and raised concerns to the development team that the prediction model could be skewed/inaccurate. The histogram reassures the team that the dataset was well-balanced and could be trusted to train the prediction model. The histogram is displayed below:



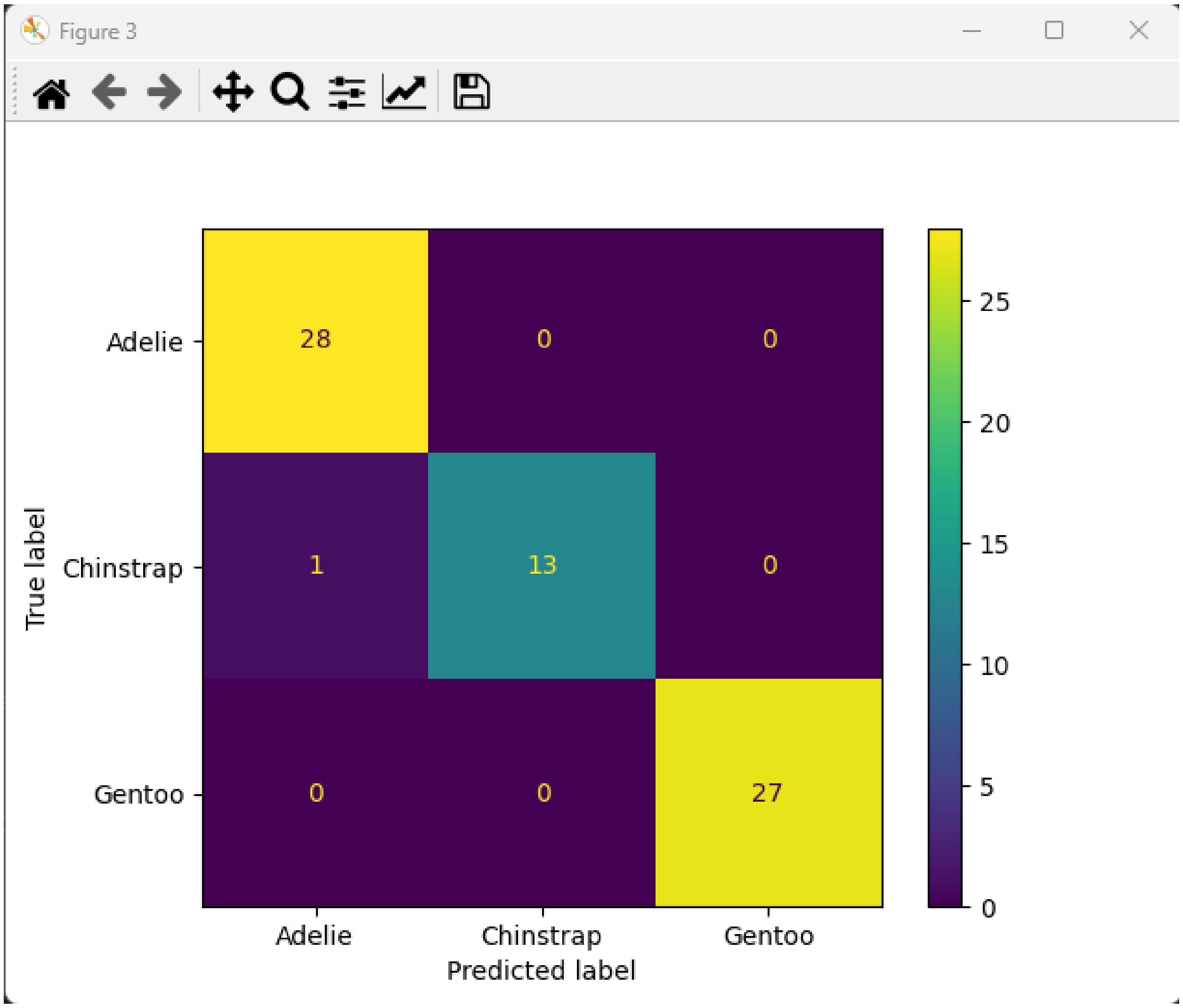
In this chart, we can see that the Adelie has the greatest number of entries about ~145, and the Gentoo trails close behind at ~121. Meanwhile, the Chinstrap has the least number of entries at ~70. Because the Chinstrap is not dramatically small (0-40 range), we can confidently rely on this dataset to represent all three species well.

The second visualization is the scatter matrix. The scatter matrix shows the relationship between different penguin measurements. The scatter matrix allowed the development team and zoologists to observe the relationship between different penguin dimensions (culmen length, culmen depth, flipper length, and body mass). The scatter matrix plots these measurements against each other to show the correlation between each measurement and enabled the user to determine correlations such as flipper length growing alongside body mass (which makes sense). For the developer, the scatter matrix ensures that only relevant data is processed while developing the model. If there is a clear missing correlation, the developer could reconsider whether the data was truly relevant to the model. The developer would then need to discuss this with the team and client, hopefully during the Requirements or Design phase. The scatter matrix is displayed below:



As an interesting note, 6 of the 12 scatter graphs show 2 separate correlation lines, which likely indicate differences in species development, such as one species having genetically smaller culmen depth compared to another species despite being at the same body mass. In all graphs, there is a strong positive correlation between all dimensions. This makes sense since as a penguin grows, we can expect all parts of the penguin to grow somewhat linearly.

The last visualization is the Confusion Matrix. We created the Confusion Matrix to visualize the accuracy of the model's prediction from the "scikit-learn" module. The confusion matrix displays all the correctly predicted penguins (on the diagonal) and the incorrectly predicted penguins (everywhere else). The confusion matrix assures the user and assured the development team that the prediction model can accurately predict penguin species because it has reserved 20% of the dataset as test data. The prediction model will use the remaining 80% of the dataset to train the model. As detailed by our requirements, we're requiring a 90% accuracy rate, and at any point, if the accuracy rate falls below 90%, we will have to return to the beginning of the waterfall methodology to seek a solution. The confusion matrix is shown below:



True label \ Predicted label	Adelie	Chinstrap	Gentoo
Adelie	28	0	0
Chinstrap	1	13	0
Gentoo	0	0	27

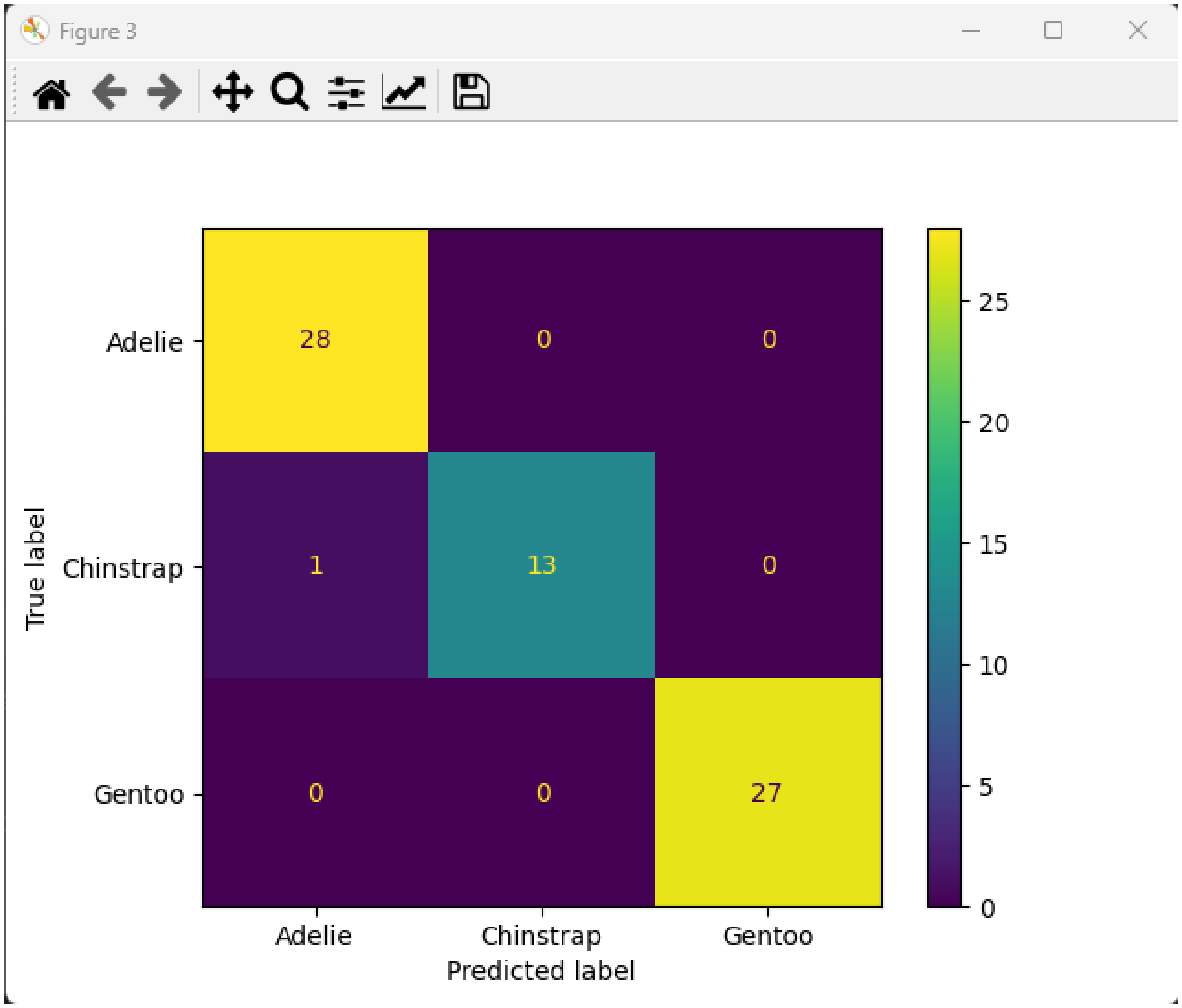
In this confusion matrix, we observe that the prediction model predicted Adelie correctly 28 times, Chinstrap correctly 13 times, and Gentoo correctly 27 times. More importantly, the model was incorrect in predicting Adelie once, when the actual penguin was a Chinstrap. Fantastically, despite the smaller sample size as noted in the histogram section, the prediction model was spot on when predicting Chinstrap, not mistaking any Chinstrap penguins for Adelie or Gentoo. Because this model was only incorrect once out of 69 tests, boasting a 98.5% accuracy rate, we can confidently say our hypothesis was met.

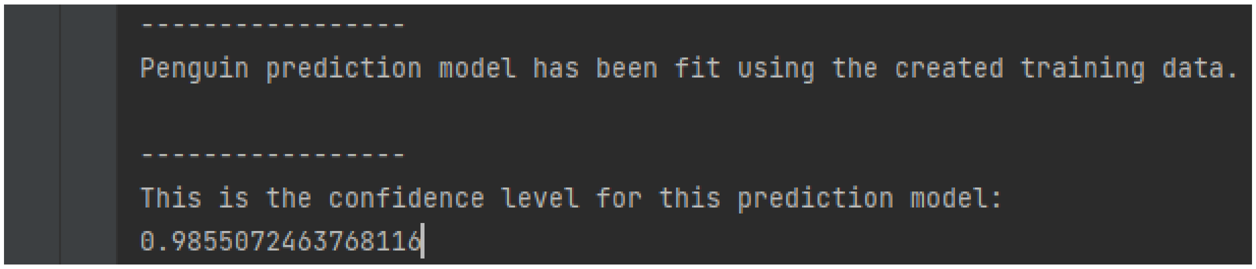
Accuracy Analysis

The metric used to assess the model is sourced from the "scikit-learn's" module. The method used is metrics.accuracy_score(), which outputs an accuracy score between 0.0-1.0 representing a percentage. Additionally, we also use the Confusion Matrix to visualize the accuracy of the prediction model.

When creating the prediction model, we segment 80% of the dataset as training data. This data will be fed and fit into the prediction model. Then, we use the remaining 20% of the dataset, segmented off for testing data, to measure the accuracy of the Logistic Regression model. Using the testing data, we can produce an accurate number to score our prediction model.

The following is an example of the application being run and a prediction model trained, fit, and assessed:





```
-----------------
Penguin prediction model has been fit using the created training data.

-----------------
This is the confidence level for this prediction model:
0.9855072463768116
```

In this example, the prediction model was assessed using the metrics.accuracy_score() method which returned an accuracy score of .985. This score is amazing because we are simply looking for an accuracy rate of 90% or higher. Furthermore, we can dive deeper into the data by looking at the visualization. The visualization displays exactly what the prediction model got correctly and incorrectly. The model predicted incorrectly Adelie once when the correct species was Chinstrap. Out of the 69 predictions, the model was spot on for 68 of them.

Application Testing

During development and near its completion, the program was executed hundreds of times to validate that the accuracy remained above 90%. Thanks to the fantastic dataset, not once did the prediction model fall below 90%. Because of the great accuracy of the Logistic Regression model, it eliminated the need to revisit the Requirements and Design phase of the waterfall methodology to define a different prediction model.

Application Files

The only file you will need is the 964HiepPham.zip file downloaded from the submission. Once downloaded, you may unzip the file and follow the instructions in the User Guide to execute the program. The User Guide is in the next section below.

User Guide

The user guide is as follows:

INSTALLATION

1. Install the latest version of Python at https://www.python.org/downloads/. Once on the page, click the "Download" button in the center of the page. Follow the directions for other Operating Systems if using an OS other than Windows. Follow the instructions from the installation package and complete the installation.

2. Install the latest version of PyCharm Community Edition at https://www.jetbrains.com/pycharm/download/. Once on the page, scroll down until you find "PyCharm Community Edition", and click "Download". Follow the installations from the installation package and complete the installation.

3. Open PyCharm

4. Search for the menu "Open". This is where we will open the Penguin Identifier program.

5. Navigate to the directory where you have saved the data product. The data product should be a folder named "964HiepPham", or whatever you changed the name to when you downloaded it.

6. Highlight the package and click "Open" in the window.

7. The project has now opened, but we are not done. We need to install modules to ensure the program runs correctly. Click on "File" in the top left corner.

8. Click the drill down on "Project: 964HiepPham"

9. Click on Python Interpreter. This is where we will install the required modules to run the program.

10. Within the Python Interpreter menu, click on the "+"

11. In the search, type "pandas", and install the exact module highlighted by clicking "Install Package" at the bottom. The installation may take a while, depending on computer

specifications. You can see the progress at the bottom of the PyCharm IDE or wait for the green bar at the bottom of the “Available Packages” menu that says “Package ‘XYZ’ installed successfully”.

12. In the search, type “matplotlib”, and install the exact module highlighted by clicking “Install Package” at the bottom. The installation may take a while, depending on computer specifications. You can see the progress at the bottom of the PyCharm IDE or wait for the green bar at the bottom of the “Available Packages” menu that says “Package ‘XYZ installed successfully”.

13. In the search, type “scikit-learn”, and install the exact module highlighted by clicking “Install Package” at the bottom. The installation may take a while, depending on computer specifications. You can see the progress at the bottom of the PyCharm IDE or wait for the green bar at the bottom of the “Available Packages” menu that says “Package ‘XYZ installed successfully”.

14. Once all 3 modules have been installed, the INSTALLATION part is complete! Please continue to “USING THE APPLICATION”

USING THE APPLICATION

1. Ensure you do not modify the code at any point.

2. Click on the green arrow on the top right of the PyCharm IDE. This will run the application.

3. A console window will appear from the bottom of your screen showing output. It is recommended you drag the top of the console window up to enlarge the console, and to prevent code modification.

4. The console will display exactly what the code is doing. If you would like to know exactly what the code has done, we recommend enlarging the console and reading the progress steps.

5. Three plots will appear on your screen. This is for your situational awareness. Figure 1 shows the count of each species in the dataset. Figure 2 displays the scatter matrix showing the relationship between the penguin dimensions. Figure 3 displays the Confidence Matrix, showing

how accurate the prediction model is (numbers on the diagonal means the model predicted correctly).

6. To continue with the application, close all three plots. The console also directs you to close all graphs.

7. The Penguin Identifier will also show you the confidence level for the prediction model, ranging from 0.0 to 1.0. If at any point, you see the model output a confidence level of less than the agreed-upon .90, please contact us immediately. Though, in our hundreds of test runs, we've never seen the confidence level dip below .90.

8. The program will ask you to enter '1' to continue. Please continue only if you are satisfied with the confidence level or want to continue identifying penguin species. Otherwise, enter '2' to quit the program.

9. The Penguin Identifier instructs you to provide the following measurements in order separated by a space: culmen length in millimeters, culmen depth in millimeters, flipper length in millimeters, and body mass in grams. The console also displays an example and the corresponding prediction.

10. The Penguin Identifier will output the predicted penguin species based on the inputted dimensions.

11. The Penguin Identifier will loop back and ask you again to (1) continue, or (2) quit.

12. Once you have completed identifying all your penguins, you may enter '2' to quit. The program is complete, and you may close out PyCharm.

13. Last reminder, please do not modify the code.

Summation of Learning Experience

I had zero machine-learning experience before this project. Dr. Jim Ashe taught me all I needed to know about machine learning for the creation of this project. Dr. Jim Ashe walked through the entire process of pulling data, creating a prediction model, fitting & testing the model, creating visualizations, and using the said model to predict a value. Dr. Jim Ashe was instrumental in my learning, and I feel confident in creating another machine-learning model using a different dataset if a task required it. Creating this machine learning project deepened my understanding of all machine learning applications out there in the world, and I'm able to grasp the concepts stronger than before this course. I have cited videos Dr. Jim Ashe sent me that greatly aided me in the development of this project, along with the C964 Capstone Home page.

References

As instructed in the Kaggle dataset at https://www.kaggle.com/datasets/parulpandey/palmer-archipelago-antarctica-penguin-data, I am citing the data using the requested citation (Gorman KB):

Ashe, Jim. “Dr. Jim Ashe C964 Webex Guide.” Webex, wgu.webex.com/recordingservice/sites/wgu/recording/9469df1f7d63103abf7f0050568114a0/playback. Accessed 25 July 2023.

C964 Course Faculty Team. “Computer Science Capstone.” Welcome to C964! - Computer Science Capstone, ashejim.github.io/C964/intro.html. Accessed 20 July 2023.

Gorman KB, Williams TD, Fraser WR (2014) Ecological Sexual Dimorphism and Environmental Variability within a Community of Antarctic Penguins (Genus *Pygoscelis*). PLoS ONE 9(3): e90081. doi:10.1371/journal.pone.0090081

Urton, James. “To Tell the Sex of a Galápagos Penguin, Measure Its Beak, Researchers Say.” UW News, www.washington.edu/news/2018/06/27/to-tell-the-sex-of-a-galapagos-penguin-measure-its-beak-researchers-say/. Accessed 21 July 2023.