Rule-Based Penguins in Pandas

The objective of this task is to categorize or forecast a particular type of penguins using two characteristics accessible in the dataset. Similar to the approach we demonstrated in class on Monday, you will create your own rule-oriented classifier utilizing two threshold parameters for the selected feature.

Brandon Mitchell and Michael Hvizdos

In [1]:
```
import pandas as pd
import plotly as plt
import plotly.express as px
```

In [ ]:
```
!pip install pandas
!pip install plotly
```

In [ ]:
```
# install libraries if needed
# !pip3 install pandas --user codio
# !pip3 install plotly --user codio
!pip install packaging
!pip install seaborn
```

In [2]:
```
df = pd.read_csv("penguins.csv")
df.shape
```

Out[2]:
```
(344, 7)
```

In [3]:
```
df.head()
```

Out[3]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	female

Problem 1

Develop a Python function that receives three arguments, including two threshold values utilized by the rule-oriented classifier and the species type, such as Adelie penguin, you wish to forecast using the classifier. The function should return a Pandas dataframe as the confusion matrix output and precision and recall metrics for your model.

In [9]:
```
def rule_based_class(threshold1, threshold2, species):
    penguins = pd.read_csv('penguins.csv')
    #This makes a column that will specify the models predicted species bas
    penguins['predicted_species'] = penguins.apply(lambda row: 'Gentoo' if
    #This will make the model know which species the user wants to predict
    penguins = penguins[penguins['species'] == species]
    #True positive (where predictions are correct)
    tp = len(penguins[(penguins['species'] == species) & (penguins['predict
    #False Positives (where predictions guess the specified species but it
    fp = len(penguins[(penguins['species'] != species) & (penguins['predict
    #False Negatives (where predictions guess incorrect species even though
    fn = len(penguins[(penguins['species'] == species) & (penguins['predict
    #True Negatives (where the predictions guess it is not the specified sp
    tn = len(penguins[(penguins['species'] != species) & (penguins['predict

    #The true positives divided by the total of postives makes the precisio
    precision = tp / (tp + fp)
    #The recall determines what the ratio of correctness the model is at
    recall = tp / (tp + fn)
    #The confusion matrix will show the distribution of predictions by the
    confusion_matrix = pd.DataFrame({'Predicted ' + species: [tp, fp], 'Pre
    #returns the necessary materials for analysis
    return confusion_matrix, precision, recall
```

In [10]:
```
confusion_matrix, precision, recall = rule_based_class(4500, 200, 'Gentoo')
print('Confusion Matrix:')
display(confusion_matrix)
print('Precision:', precision)
print('Recall:', recall)
```

```
Confusion Matrix:
```

	Predicted Gentoo	Predicted Not Gentoo
Actual Gentoo	106	18
Actual Not Gentoo	0	0

```
Precision: 1.0
Recall: 0.8548387096774194
```

Problem 2

Subsequently, employing the ipwidgets interactive library, enable users to construct their models by designating specific species and thresholds. Your "interact" function must generate a plot, confusion matrix, and precision/recall metrics. Ensure that the output of the model is clearly displayed by correctly color-coding the predicted and actual values and accurately labeling the results.

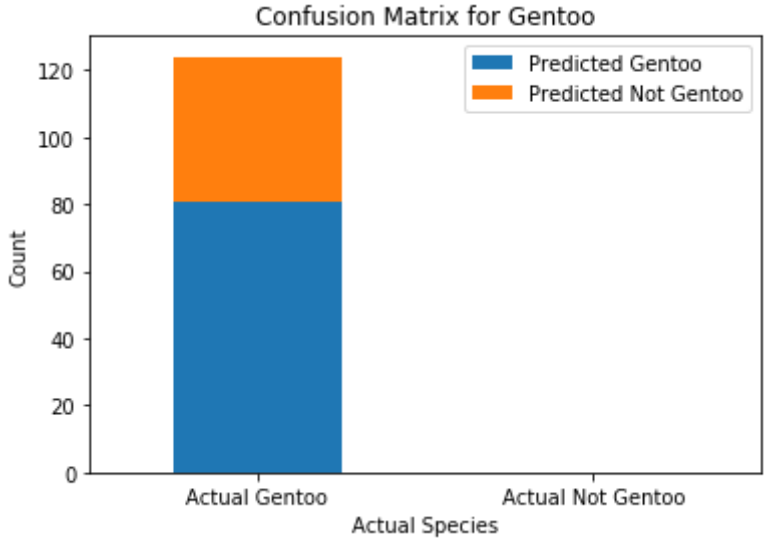
In [6]:
```
import matplotlib.pyplot as plt
import ipywidgets as widgets
from ipywidgets import interactive
def rule_based_class():
    penguins = pd.read_csv('penguins.csv')
    #creates a function that takes three inputs as an argument
    def predict_species(threshold1, threshold2, species):
        #This makes a column that will specify the models predicted species
        penguins['predicted_species'] = penguins.apply(lambda row: 'Gentoo'
        #This will make the model know which species the user wants to pred
        penguins_filtered = penguins[penguins['species'] == species]
        #True positive (where predictions are correct)
        tp = len(penguins_filtered[(penguins_filtered['species'] == species
        #False Positives (where predictions guess the specified species but
        fp = len(penguins_filtered[(penguins_filtered['species'] != species
        #False Negatives (where predictions guess incorrect species even th
        fn = len(penguins_filtered[(penguins_filtered['species'] == species
        #True Negatives (where the predictions guess it is not the specifie
        tn = len(penguins_filtered[(penguins_filtered['species'] != species
        #The true positives divided by the total of postives makes the prec
        precision = tp / (tp + fp)
        #The recall determines what the ratio of correctness the model is a
        recall = tp / (tp + fn)
        #The confusion matrix will show the distribution of predictions by
        confusion_matrix = pd.DataFrame({'Predicted ' + species: [tp, fp],
        #THESE NEXT 5 lines create a bar chart that will show the amount of
        #based on the interactive inputs the user will specify
        ax = confusion_matrix.plot(kind='bar', stacked=True, rot=0)
        ax.set_xlabel('Actual Species')
        ax.set_ylabel('Count')
        ax.set_title('Confusion Matrix for ' + species)
        plt.show()
        #These print statements show the precison recall and confusion matr
        print("Precision: {:.2f}".format(precision))
        print("Recall: {:.2f}".format(recall))
        display(confusion_matrix)

        return confusion_matrix, precision, recall
    #These create the interactable sliders in order to produce the desired
    #The max and min for the lenght and mass are specified along with which
    threshold1_slider = widgets.IntSlider(min=penguins['body_mass_g'].min()
    threshold2_slider = widgets.FloatSlider(min=penguins['bill_length_mm'].
    species_dropdown = widgets.Dropdown(options=penguins['species'].unique(
    #Makes the interactives usable
    output = interactive(predict_species, threshold1=threshold1_slider, thr
    return output
```

In [7]:
```
rule_based_class()
```

Body Mass... 4830

Bill Length ... 42.20

Penguin S... Gentoo




```
Precision: 1.00
Recall: 0.65
```

	Predicted Gentoo	Predicted Not Gentoo
Actual Gentoo	81	43
Actual Not Gentoo	0	0

In [ ]: