# Pic 16a Final Project

March 16, 2024

## 1 Pic 16a Final Project

Cynthia Du, Brandon Cho, Hongzhe Du

All three of us worked together to come up with our overall idea for the project. Cynthia and Hongzhe worked together on the data import and cleaning task , while Brandon checked it over. All three of us worked on the exploratory analysis. Brandon made the pairplot, Cynthia made the scatterplot and table, and Hongzhe made the stacked bar plots. We all worked together to do feature selection and agreed on the features we should select based on the output. All three of us worked on a model, where Hongzhe did the Neural Network, Cynthia did the Random Forest model, and Brandon did the Logistic Regression model. We also collaborated in the Decision Region plot. All three of us did the discussions of the corresponding model we worked on, and all three of us worked together to do the overal discussion of the project.

### 1.1 Data Import and Cleaning

[192]:
```python
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

penguins = pd.read_csv('/content/palmer_penguins.csv')
```

#### 1.1.1 Separate Data into Training and Test Sets

[193]:
```python
# Make "Species" column easier to interpret by only using the first word
penguins["Species"] = penguins["Species"].str.split().str.get(0)
penguins = penguins[penguins["Sex"]!="."]
```

[194]:
```python
from sklearn.model_selection import train_test_split

# target data
y = penguins["Species"]

# predictor data
X = penguins.drop(["Species"], axis = 1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=1234)
```

### 1.1.2 Data Cleaning

[195]:
```
from sklearn import preprocessing

def clean_data(data):
    '''
    Cleans the provided dataset by removing unrelated columns, handling missing␣
 ↪values,
    and encoding categorical variables to numeric format.

    Parameters:
        data (DataFrame): The dataset to clean.

    Returns:
        DataFrame: The cleaned dataset with categorical variables encoded.
        list: Indices of rows that were removed due to containing null values.
    '''
    df = data.copy()
    # drop columns unrelated to species to easier analyze data
    df = df.drop(["studyName", "Sample Number", "Stage","Region", "Individual␣
 ↪ID", "Date Egg", "Comments"], axis = 1)
    # get indices of rows containing null value
    nan_indices = df.index[df.isnull().any(axis=1)].tolist()
    # drop na values
    df = df.dropna()

    le = preprocessing.LabelEncoder()
    df["Sex"] = le.fit_transform(df["Sex"])
    df["Island"] = le.fit_transform(df["Island"])
    df["Clutch Completion"] = le.fit_transform(df["Clutch Completion"])

    return df, nan_indices
# clean test and training data separately
X_train, train_nan_indices = clean_data(X_train)
X_test, test_nan_indices = clean_data(X_test)

y_train = y_train.drop(train_nan_indices)
y_test = y_test.drop(test_nan_indices)



le = preprocessing.LabelEncoder()
y_train = pd.Series(le.fit_transform(y_train), index=y_train.index)
y_test = pd.Series(le.fit_transform(y_test), index=y_test.index)

inverse_mapping = dict(zip(range(len(le.classes_)), le.classes_))
original_class_name = inverse_mapping[0]
```

```python
print("Numeric to class name mapping:", inverse_mapping)
```

```
Numeric to class name mapping: {0: 'Adelie', 1: 'Chinstrap', 2: 'Gentoo'}
```

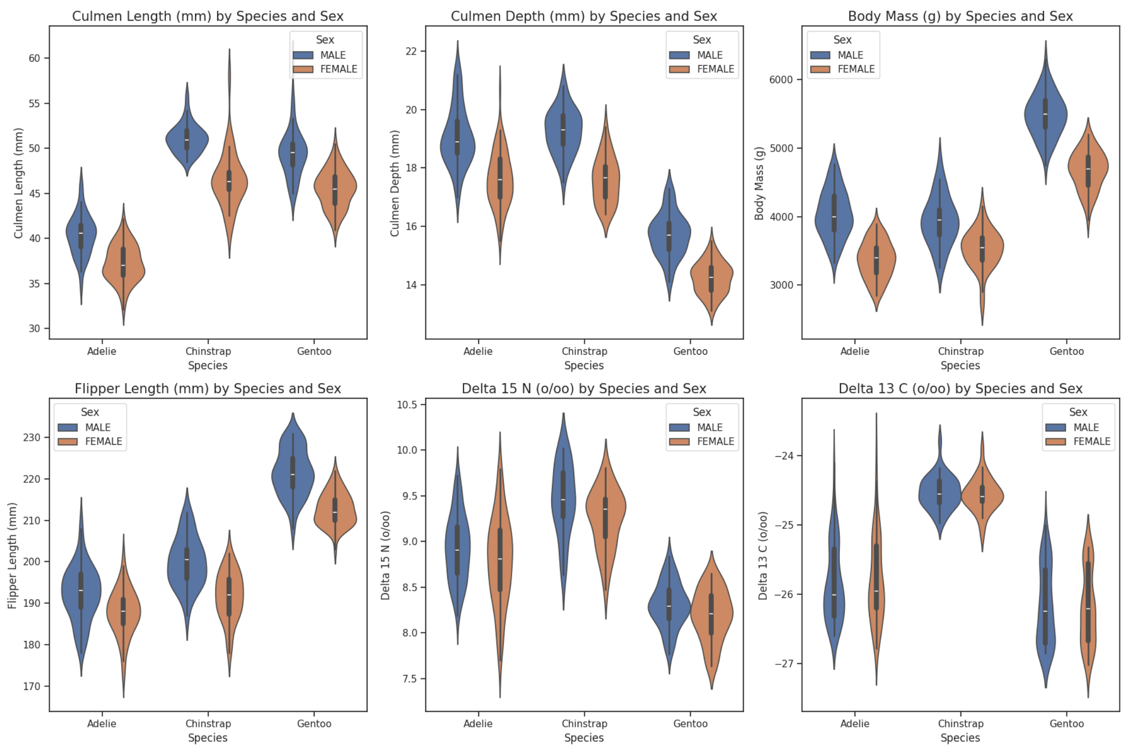
## 1.2 Exploratory Analysis

[196]:
```python
def penguin_summary_table(group_cols, value_cols):
    table = penguins.groupby(group_cols)[value_cols].mean().reset_index()
    penguins.round(2)
    return table

penguin_summary_table(["Species"], ["Culmen Length (mm)", "Culmen Depth (mm)",␣
 ↪"Body Mass (g)",
           "Flipper Length (mm)", "Delta 15 N (o/oo)", "Delta 13 C (o/oo)"])
```

[196]:
```
     Species  Culmen Length (mm)  Culmen Depth (mm)  Body Mass (g)  \
0     Adelie           38.791391          18.346358    3700.662252
1  Chinstrap           48.833824          18.420588    3733.088235
2     Gentoo           47.529508          14.976230    5077.663934

   Flipper Length (mm)  Delta 15 N (o/oo)  Delta 13 C (o/oo)
0           189.953642           8.859733         -25.804194
1           195.823529           9.356155         -24.546542
2           217.188525           8.247026         -26.185305
```

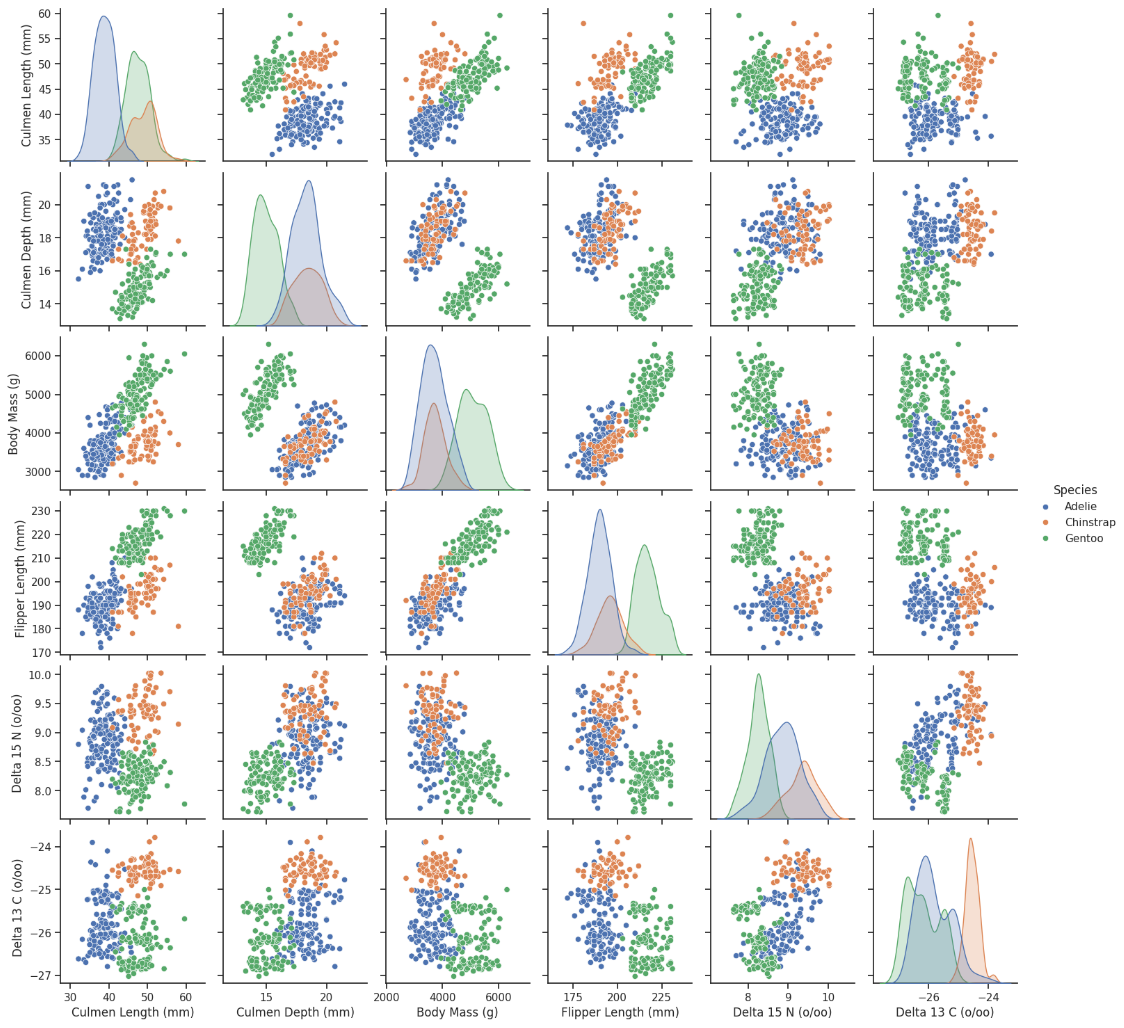
From the table, we were able to see significant enough differences in certain values to make decisions on what to compare in our study. For example, the differences in body mass and culmen length were variables we wanted to study. Furthermore, the overlap/value similarities between the different species also contributed to this.

[197]:
```python
columns = ["Culmen Length (mm)", "Culmen Depth (mm)", "Body Mass (g)",
           "Flipper Length (mm)", "Delta 15 N (o/oo)", "Delta 13 C (o/oo)"]



fig, axes = plt.subplots(2, 3, figsize=(18, 12)) # Setup the figure and axes␣
 ↪for a 2x3 grid
axes = axes.flatten()

for ax, column in zip(axes, columns):
    # Create the violin plot for the current column, grouped by species and sex
    sns.violinplot(x="Species", y=column, hue="Sex", data=penguins, ax=ax)
    ax.set_title(f"{column} by Species and Sex", size=15)

plt.tight_layout()
plt.show()
```



```
[198]: import seaborn as sns
       sns.set_theme(style="ticks")

       sns.pairplot(penguins[["Species", "Culmen Length (mm)", "Culmen Depth (mm)",␣
        ↪"Body Mass (g)", "Flipper Length (mm)", "Delta 15 N (o/oo)",         "Delta␣
        ↪13 C (o/oo)"]], hue="Species")
```

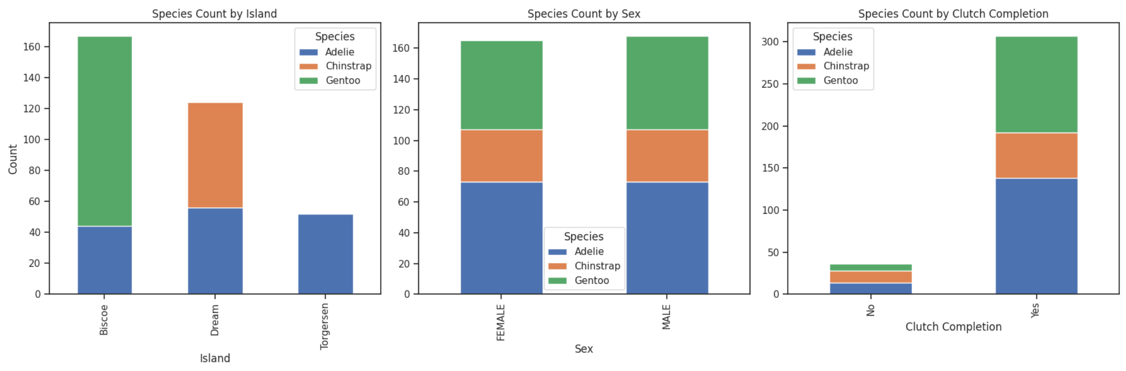
```
[198]: <seaborn.axisgrid.PairGrid at 0x7e25e1194a00>
```



The analysis of the paired scatterplots and the violin plots across various quantitative features reveals that the species Adelie and Chinstrap exhibit significant overlap in multiple plots. This observation indicates that these two species possess similar characteristics across several measured attributes, suggesting a close resemblance in their physical or ecological traits. Such overlap highlights the challenge of distinguishing between Adelie and Chinstrap penguins based solely on a broad range of quantitative features, underscoring the necessity for identifying more discriminative attributes for effective classification.

However, a distinct pattern emerges in the scatterplot comparing culmen length to culmen depth, where all three species—Adelie, Chinstrap, and Gentoo—demonstrate clear differentiation. This separation indicates that culmen length and culmen depth are notably distinct across species, with each species displaying a unique combination of these two features. The pronounced divergence in this specific plot suggests that culmen length and culmen depth are critical factors in the morphological differentiation of penguin species and thus serve as valuable features for classification purposes. Through the plots we can also see that for certain features, the male penguins exhibit differing body traits such as culmen length and body mass. We can make the assumption that this is because they are larger in average and have bigger bodies than the female penguins.

[199]:
```
fig, axes = plt.subplots(1, 3, figsize=(18, 6))



# Plotting for 'Island'
island_species_counts = penguins.groupby(['Island', 'Species']).size().
 ↪unstack(fill_value=0)
island_species_counts.plot(kind='bar', stacked=True, ax=axes[0])
axes[0].set_title('Species Count by Island')
axes[0].set_xlabel('Island')
axes[0].set_ylabel('Count')


# Repeat the aggregation and plotting for 'Sex'
sex_species_counts = penguins.groupby(['Sex', 'Species']).size().
 ↪unstack(fill_value=0)
sex_species_counts.plot(kind='bar', stacked=True, ax=axes[1])
axes[1].set_title('Species Count by Sex')
axes[1].set_xlabel('Sex')


# Repeat the aggregation and plotting for 'Clutch Completion'
clutch_species_counts = penguins.groupby(['Clutch Completion', 'Species']).
 ↪size().unstack(fill_value=0)
clutch_species_counts.plot(kind='bar', stacked=True, ax=axes[2])
axes[2].set_title('Species Count by Clutch Completion')
axes[2].set_xlabel('Clutch Completion')

plt.tight_layout()
plt.show()
```



Based on the analysis of the stacked bar plots for each categorical variable (“Island”, “Sex”, “Clutch

Completion"), it's evident that the proportions of each penguin species within the categories of sex and clutch completion remain consistent. This uniformity across categories suggests that sex and clutch completion are not significant factors in predicting penguin species. Such an observation indicates that these variables might have limited predictive power regarding the species differentiation in our dataset.

Conversely, the "Species Count by Island" plot reveals a distinct pattern: both Gentoo and Chinstrap penguins are predominantly found on specific islands. This exclusivity of habitat demonstrates a strong correlation between the island variable and penguin species, suggesting that the island is a significant predictor of penguin species. The geographical isolation provided by islands likely plays a crucial role in the distribution of different penguin species, making it a valuable parameter for species identification in predictive models.

## 1.3 Feature Selection

[200]:
```
# perform analysis to choose one qualitative, two quantitative variables.␣
↪justify your choice

from sklearn import preprocessing
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from itertools import combinations
```

[201]:
```
def check_col_score(cols):
    """
    Trains and evaluates a model via cross validation on the cols features
    Returns the CV mean score

    Parameters:
        cols (list): list of parameters.

    Returns:
        DataFrame: The cleaned dataset with categorical variables encoded.
        Integer: CV score of the model using selected features
    """

    # initialize logistic regression model, get cross validation score
    LR = LogisticRegression(max_iter=3000)
    return cross_val_score(LR, X_train[cols], y_train, cv = 5).mean()

# list of availible features
qualitative = ["Island", "Sex", "Clutch Completion"]
quantitative = ["Culmen Length (mm)", "Culmen Depth (mm)", "Body Mass (g)",␣
↪"Flipper Length (mm)", "Delta 15 N (o/oo)",        "Delta 13 C (o/oo)"]

# creating all possible one qual two quan combos
combos = []
```

```
for qual in qualitative:
  for quan in list(combinations(quantitative, 2)):
    combo = [qual, quan[0], quan[1]]
    combos.append(combo)

#get cross validation scores of each combo
scores = []
for cols in combos:
    scores.append(check_col_score(cols))

data = {"Col Combos": ["; ".join(combo) for combo in combos], "Score": scores}
df_table = pd.DataFrame(data)
df_table
```

[201]:

```
                                           Col Combos     Score
0       Island; Culmen Length (mm); Culmen Depth (mm)  0.976697
1           Island; Culmen Length (mm); Body Mass (g)  0.961388
2     Island; Culmen Length (mm); Flipper Length (mm)  0.969005
3       Island; Culmen Length (mm); Delta 15 N (o/oo)  0.969005
4       Island; Culmen Length (mm); Delta 13 C (o/oo)  0.961086
5            Island; Culmen Depth (mm); Body Mass (g)  0.794570
6      Island; Culmen Depth (mm); Flipper Length (mm)  0.833183
7        Island; Culmen Depth (mm); Delta 15 N (o/oo)  0.817722
8        Island; Culmen Depth (mm); Delta 13 C (o/oo)  0.926471
9          Island; Body Mass (g); Flipper Length (mm)  0.709578
10           Island; Body Mass (g); Delta 15 N (o/oo)  0.763801
11           Island; Body Mass (g); Delta 13 C (o/oo)  0.767647
12     Island; Flipper Length (mm); Delta 15 N (o/oo)  0.872021
13     Island; Flipper Length (mm); Delta 13 C (o/oo)  0.949623
14       Island; Delta 15 N (o/oo); Delta 13 C (o/oo)  0.899095
15         Sex; Culmen Length (mm); Culmen Depth (mm)  0.988386
16             Sex; Culmen Length (mm); Body Mass (g)  0.972926
17       Sex; Culmen Length (mm); Flipper Length (mm)  0.976848
18         Sex; Culmen Length (mm); Delta 15 N (o/oo)  0.965083
19         Sex; Culmen Length (mm); Delta 13 C (o/oo)  0.980694
20              Sex; Culmen Depth (mm); Body Mass (g)  0.794570
21        Sex; Culmen Depth (mm); Flipper Length (mm)  0.844721
22          Sex; Culmen Depth (mm); Delta 15 N (o/oo)  0.821719
23          Sex; Culmen Depth (mm); Delta 13 C (o/oo)  0.945701
24            Sex; Body Mass (g); Flipper Length (mm)  0.763348
25              Sex; Body Mass (g); Delta 15 N (o/oo)  0.794796
26              Sex; Body Mass (g); Delta 13 C (o/oo)  0.813801
27        Sex; Flipper Length (mm); Delta 15 N (o/oo)  0.879789
28        Sex; Flipper Length (mm); Delta 13 C (o/oo)  0.949472
29          Sex; Delta 15 N (o/oo); Delta 13 C (o/oo)  0.867949
30  Clutch Completion; Culmen Length (mm); Culmen …  0.961388
31  Clutch Completion; Culmen Length (mm); Body Ma…  0.961237
```

```
32  Clutch Completion; Culmen Length (mm); Flipper…  0.965158
33  Clutch Completion; Culmen Length (mm); Delta 1…  0.937934
34  Clutch Completion; Culmen Length (mm); Delta 1…  0.937858
35  Clutch Completion; Culmen Depth (mm); Body Mas…  0.783032
36  Clutch Completion; Culmen Depth (mm); Flipper …  0.825189
37  Clutch Completion; Culmen Depth (mm); Delta 15…  0.798492
38  Clutch Completion; Culmen Depth (mm); Delta 13…  0.926395
39  Clutch Completion; Body Mass (g); Flipper Leng…  0.678582
40  Clutch Completion; Body Mass (g); Delta 15 N (…  0.756033
41  Clutch Completion; Body Mass (g); Delta 13 C (…  0.728808
42  Clutch Completion; Flipper Length (mm); Delta …  0.860633
43  Clutch Completion; Flipper Length (mm); Delta …  0.953544
44  Clutch Completion; Delta 15 N (o/oo); Delta 13…  0.864027
```

[202]:
```python
# get the combo of highesr CV scores

X_train_lr = X_train[["Sex", "Culmen Length (mm)", "Body Mass (g)"]]
X_test_lr = X_test[["Sex", "Culmen Length (mm)", "Body Mass (g)"]]
df_table.loc[df_table['Score'].idxmax()]
```

[202]:
```
Col Combos    Sex; Culmen Length (mm); Culmen Depth (mm)
Score                                           0.988386
Name: 15, dtype: object
```

For using the logistic regression model, we selected feature using cross validation on all 45 possible feature combos. By comparing the CV score of each combo, we choose the best perform combo which is 'Sex', 'Culmen Length (mm)', and 'Body Mass (g).'

## 1.4 Modeling

The Numeric to class name mapping used in the following section: {0: 'Adelie', 1: 'Chinstrap', 2: 'Gentoo'}

This is the reusable decision boundary plotting function

[203]:
```python
from matplotlib.colors import ListedColormap
contour_cmap = ListedColormap(['#98FA91', '#F086A9', '#9B8EF3']) # Light colors␣
 ↪for the decision surface
scatter_cmap = ListedColormap(['#1AF00B', '#F53575', '#0000FF']) # More␣
 ↪saturated colors for the points

def plot_decision_regions(X, y, qual_feature, quantitative_features,␣
 ↪classifier):
    '''
    Plots decision regions for a classifier on a 2D projection, separated by a␣
 ↪qualitative feature.

    Parameters:
```

```
    - X (pd.DataFrame): The input features, with both qualitative and␣
↪quantitative features.
    - y (array-like): The target variable, with class labels.
    - qual_feature (str): The name of the qualitative feature in the dataset␣
↪used to separate the plots.
    - quantitative_features (list of str): The names of the two quantitative␣
↪features used for plotting the decision regions.
    - classifier: The trained classifier object that has a .predict() method.␣
↪It is used to predict the classes
      across the decision region for each subset defined by the qualitative␣
↪feature.
    '''
    global_x0_min, global_x0_max = X[quantitative_features[0]].min(),␣
↪X[quantitative_features[0]].max()
    global_x1_min, global_x1_max = X[quantitative_features[1]].min(),␣
↪X[quantitative_features[1]].max()
    unique_classes = np.unique(y)
    color_dict = {unique_classes[i]: scatter_cmap.colors[i % len(scatter_cmap.
↪colors)] for i in range(len(unique_classes))}

    fig, axarr = plt.subplots(nrows=1, ncols=len(X[qual_feature].unique()),␣
↪sharex='col', sharey='row', figsize=(len(X[qual_feature].unique())*5, 4))

    if len(X[qual_feature].unique()) == 1:
        axarr = [axarr]

    grid_x = np.linspace(global_x0_min, global_x0_max, 501)
    grid_y = np.linspace(global_x1_min, global_x1_max, 501)
    xx, yy = np.meshgrid(grid_x, grid_y)

    for idx, val in enumerate(X[qual_feature].unique()):
        filter_idx = X.index[X[qual_feature] == val].tolist()
        X_filtered = X[X[qual_feature] == val]
        y_filtered = y[filter_idx]

        XX = xx.ravel()
        YY = yy.ravel()
        XY = pd.DataFrame({
            qual_feature: np.repeat(val, XX.size),
            quantitative_features[0]: XX,
            quantitative_features[1]: YY
        })
        p = classifier.predict(XY)
        p = p.reshape(xx.shape)

        ax = axarr[idx]
        contour = ax.contourf(xx, yy, p, alpha=0.4, cmap=contour_cmap)
        for class_label in unique_classes:
            ax.scatter(X_filtered[quantitative_features[0]][y_filtered ==␣
↪class_label],
                       X_filtered[quantitative_features[1]][y_filtered ==␣
↪class_label],
                       color=color_dict[class_label], label=f'Class␣
↪{class_label}', edgecolor='k', s=20)
        ax.set(xlabel=quantitative_features[0], ylabel=quantitative_features[1])

        ax.set_title(f"{qual_feature}: {val}")

    # Add a single legend outside the rightmost subplot
    handles, labels = axarr[0].get_legend_handles_labels()
    fig.legend(handles, labels, loc='upper right', bbox_to_anchor=(1.1, 1))
    plt.tight_layout()
    plt.show()
```

###Nerual Network

[204]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,␣
↪accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import f_classif
```

Since training NNs on all possible feature combos is too computatinoally expensive, We will use ANOVA to select the best combo features for this model.

[205]:
```
# Select the best features based on ANOVA F-value
results = []
for combo in combos:
  X_subset = X_train[combo]
  f_values, p_values = f_classif(X_subset, y_train)
  results.append((np.mean(f_values), combo))

results.sort(reverse=True, key=lambda x: x[0])
best_features = results[0][1]
X_train_nn = X_train[best_features]
X_test_nn = X_test[best_features]
best_features
```

[205]:
```
['Island', 'Culmen Length (mm)', 'Flipper Length (mm)']
```

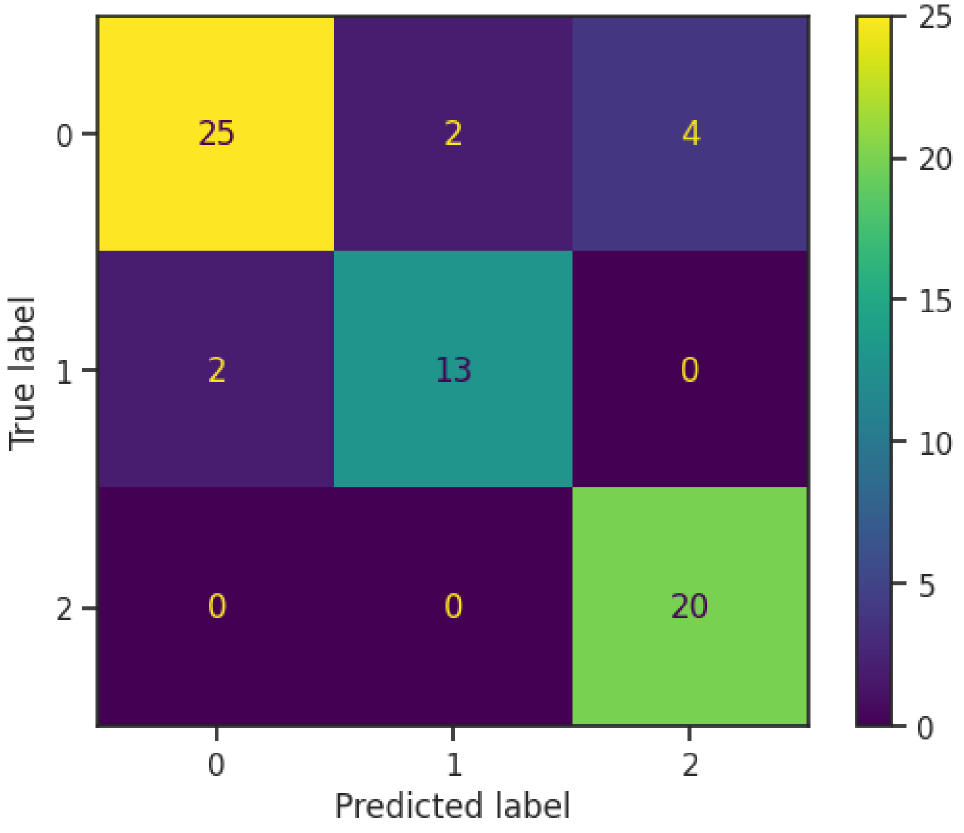
Cross validation is then used to select the best hyper parameters

[206]:
```
mlp = MLPClassifier(max_iter=1000)
parameter_space = {
    'hidden_layer_sizes': [(50,), (100,), (50,50), (100,100)],
    'activation': ['tanh', 'relu'],
    'solver': ['sgd', 'adam'],
    'alpha': [0.0001, 0.05],
}
clf = GridSearchCV(mlp, parameter_space, n_jobs=-1, cv=5)
clf.fit(X_train_nn, y_train)
print("Best parameters found:\n", clf.best_params_)
```

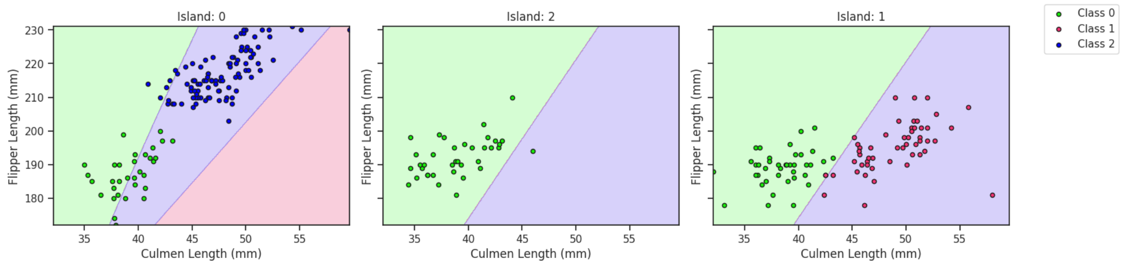
```
Best parameters found:
 {'activation': 'tanh', 'alpha': 0.05, 'hidden_layer_sizes': (100,), 'solver':
'adam'}
```

[ ]:
```
y_pred = clf.predict(X_test_nn)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

```
Accuracy: 0.88
```



```
[207]: plot_decision_regions(X_train_nn, y_train, "Island", ['Culmen Length (mm)',␣
         ↪'Flipper Length (mm)'], clf)
```

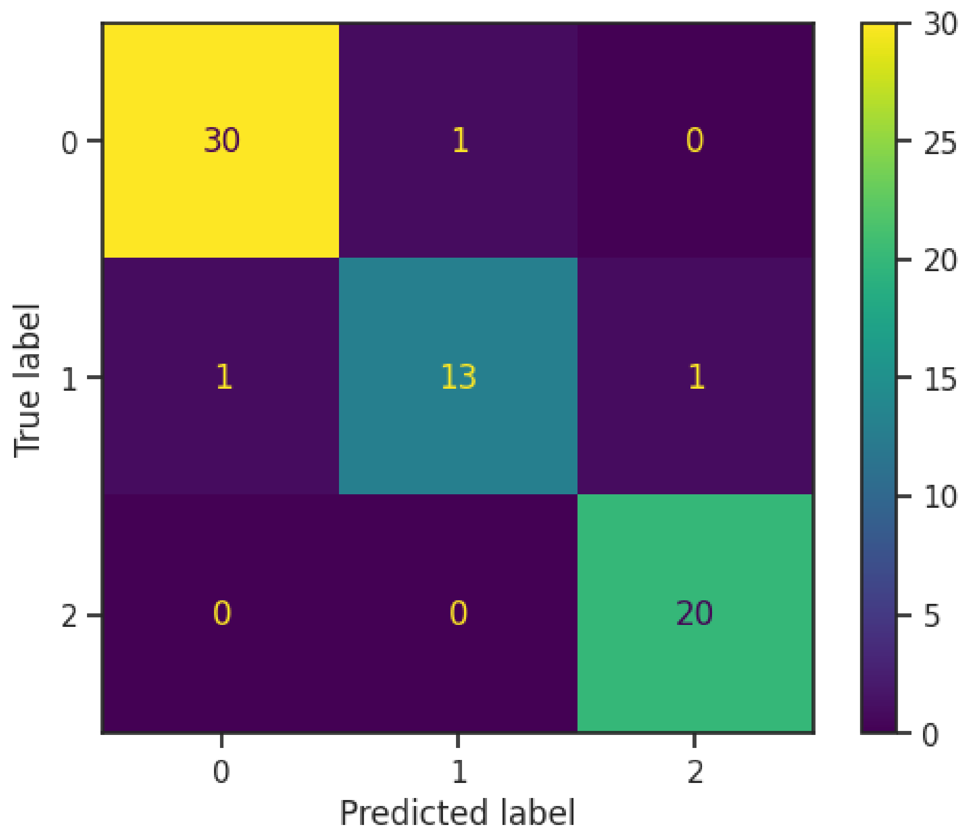


From our analysis, we believe that the largest mistake neural network models make is its difficulty in accurately handling qualitative variables. When dealing with the qualitative variables in our study, like categorical features such as island, neural networks may struggle to capture nuances accurately, even after converting them into numerical representations. This can result in reduced model accuracy due to information loss. Furthermore, due to the small size of our data set, this can also lead to some mistakes by the model.

### 1.4.1 Random Forest Model

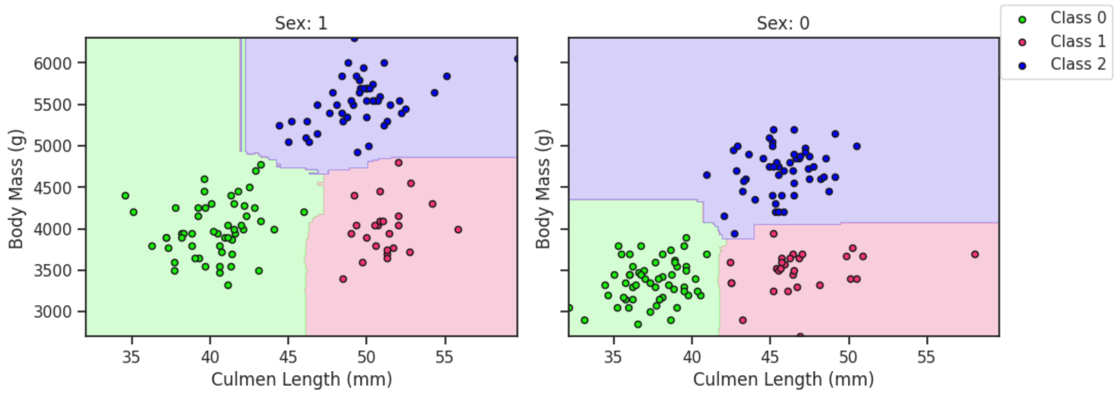
[208]:
```
# import modules needed for Random Forest
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report

# train the random forest model
rf = RandomForestClassifier(n_estimators = 100, random_state = 42)
rf.fit(X_train_lr, y_train)

# make predictions on the testing data
y_pred = rf.predict(X_test_lr)

# evaluate the model's performance
print("Accuracy:", accuracy_score(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()

print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.9545454545454546
```



```
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        31
           1       0.93      0.87      0.90        15
           2       0.95      1.00      0.98        20

    accuracy                           0.95        66
   macro avg       0.95      0.94      0.95        66
weighted avg       0.95      0.95      0.95        66
```

[209]:
```
plot_decision_regions(X_train_lr, y_train, "Sex", ["Culmen Length (mm)", "Body␣
↪Mass (g)"], rf)
```



Based on the classification report and the confusion matrix, the mistakes made by the Random Forest model can be analyzed for each species. The model demonstrates relatively high precision but lower recall for Adelie penguins. While it performs well in correctly identifying Adelie penguins, it misses some instances of this species, leading to a lower recall score. For Chinstrap penguins, the model also shows good precision and recall, although not as high as for Adelie. It still struggles to correctly classify all instances of this species, resulting in some misclassifications or missed instances. For Gentoo penguins, the model has higher recall but lower precision compared to the other species. This indicates that while it correctly identifies most Gentoo penguins, it also misclassifies some instances of other species as Gentoo.

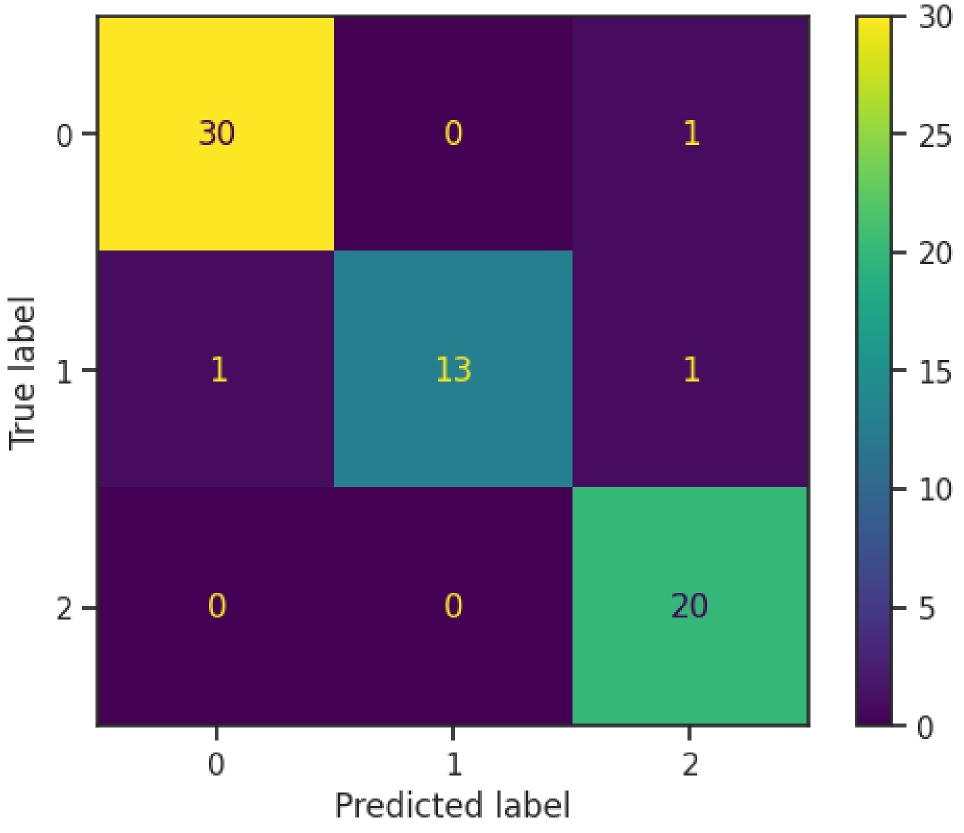
The overall model of the performance is relatively good, but it is important to understand why mistakes might occur. In the graphs and pairplots above, it is clear that the Chinstrap species is very similar to both the Adelie and Gentoo species, and the features that differentiate between Chinstrap and other penguin species are subtle or overlap in many instances. This leads to misclassifications, as Chinstrap penguins share many features as Adelie and Gentoo species. Furthermore, the small size of the dataset could have led to some more mistakes in our model.

### 1.4.2 Logistic Regression Model

[210]:
```
# fit the logistic regression model on the training data
lr = LogisticRegression(max_iter=4000)
lr.fit(X_train_lr, y_train)

# make predictions on the testing data
y_pred = lr.predict(X_test_lr)
y_pred_probs = lr.predict_proba(X_test_lr)[:, 1]

# evaluate the model's performance
print("Accuracy:", accuracy_score(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
```

```
plt.show()

print("Classification Report:\n", classification_report(y_test, y_pred))
```

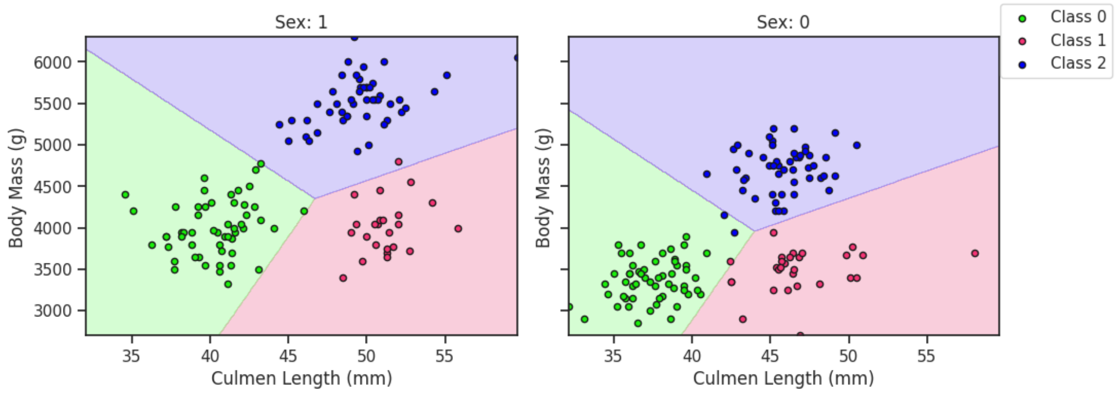
```
Accuracy: 0.9545454545454546
```



```
Classification Report:
               precision    recall  f1-score   support

           0       0.97      0.97      0.97        31
           1       1.00      0.87      0.93        15
           2       0.91      1.00      0.95        20

    accuracy                           0.95        66
   macro avg       0.96      0.94      0.95        66
weighted avg       0.96      0.95      0.95        66
```

The logistic regression model performs well for all three species, with high precision and recall for all three. This indicates that the model correctly identifies the majority of instances belonging to each species without many false positives or false negatives. F

[211]:
```
plot_decision_regions(X_train_lr, y_train, "Sex", ["Culmen Length (mm)", "Body␣
↪Mass (g)"], lr)
```



The logistic regression model performs well for all three species, with high precision and recall for all three. This indicates that the model correctly identifies the majority of instances belonging to each species without many false positives or false negatives. As shown in the decision region below, the decision boundaries are pretty well separated, with the exception of a few data points. The recall score for class 1 is relatively lower, which indicates that the decision boundary for Class 1 might be less well-defined compared to Class 0, resulting in some instances being misclassified. Some of these errors could be due to the fact that the species are relatively similar, and thus the decision boundary is less clear / overlaps with other species. However, despite this, the model performs relatively well and is able to accurately classify the species.

## 1.5 Discussion

In this case, our group had originally come to the conclusion that the neural network model would be the best model to use with our specific data set since we are comparing many different categories in this study. However, after looking at the models and comparing the accuracies, we saw that the logistic regression model and random forest would be the best models to use because it has the highest accuracies. Overall, these two models performed the best, but we believe that with an increase in the data set would result in neural network having an advantage over the other two models. Furthermore, as seen from our feature selections section, the best combo of variables to use would be Sex, Culmen Length (mm), and Body Mass (g).