Final_Project

June 2, 2022

1 Group Contributions Statement.

All three of us worked on the data acquisition and preparation. We all wrote the code to clean the data and in the discussion of how to do so, how to deal with missing values, what columns to keep, what columns to recode etc.

The data visualization was a group effort as well. All 3 of us coded at least 1 plot or table, and the discussion of each plot was done by all 3 of us.

The feature selection section was done by all 3 of us. We all worked together to discuss the best approach to take for choosing features of the model. The code for the cross validation score function was mainly done by Chen and Lucas.

We all worked together to fit the models and plot the decision regions, as well as analyzing the mistakes of each model, the confusion matrices and the decision regions. The decision regions plots were mainly coded by Ethan, and we all 3 worked on fitting the models and getting the confusion matrices, and the discussion of each model.

The overall dicussion was done as a group as well. We all 3 talked about the scoring and the mistakes made by each model, how to improve, and overfitting. We also chose which model to use as a group.

2 Data import and cleaning.

In this part, we will read in our data; separate it into training and test sets; and clean it by our defined function.

[1]:
```
# standard imports
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
```

[2]:
```
url = "https://philchodrow.github.io/PIC16A/datasets/palmer_penguins.csv"
penguins = pd.read_csv(url)
```

[3]:
```
penguins.head()
```

[3]:
```
  studyName  Sample Number                              Species  Region  \
0   PAL0708              1  Adelie Penguin (Pygoscelis adeliae)  Anvers
1   PAL0708              2  Adelie Penguin (Pygoscelis adeliae)  Anvers
```

```
2   PAL0708            3  Adelie Penguin (Pygoscelis adeliae)  Anvers
3   PAL0708            4  Adelie Penguin (Pygoscelis adeliae)  Anvers
4   PAL0708            5  Adelie Penguin (Pygoscelis adeliae)  Anvers

      Island              Stage Individual ID Clutch Completion  Date Egg  \
0  Torgersen  Adult, 1 Egg Stage          N1A1               Yes  11/11/07
1  Torgersen  Adult, 1 Egg Stage          N1A2               Yes  11/11/07
2  Torgersen  Adult, 1 Egg Stage          N2A1               Yes  11/16/07
3  Torgersen  Adult, 1 Egg Stage          N2A2               Yes  11/16/07
4  Torgersen  Adult, 1 Egg Stage          N3A1               Yes  11/16/07

   Culmen Length (mm)  Culmen Depth (mm)  Flipper Length (mm)  Body Mass (g)  \
0                39.1               18.7                181.0         3750.0
1                39.5               17.4                186.0         3800.0
2                40.3               18.0                195.0         3250.0
3                 NaN                NaN                  NaN            NaN
4                36.7               19.3                193.0         3450.0

      Sex  Delta 15 N (o/oo)  Delta 13 C (o/oo)  \
0    MALE                NaN                NaN
1  FEMALE            8.94956          -24.69454
2  FEMALE            8.36821          -25.33302
3     NaN                NaN                NaN
4  FEMALE            8.76651          -25.32426

                        Comments
0  Not enough blood for isotopes.
1                             NaN
2                             NaN
3              Adult not sampled.
4                             NaN
```

2.1 Train-test split

Now that we have our data we want to clean it. We are required to split the data before cleaning it so that information from our training set doesnt accidentally pollute the test set.

[4]:
```python
from sklearn.model_selection import train_test_split

np.random.seed(429)
# split the data
train, test = train_test_split(penguins, test_size = 0.2)
```

Next is the function to clean the data. We decided to drop the columns `studyName`, `Comments`, `Individual ID`, `Date Egg`, and `Stage`, since these columns are likely unrelated to species. The `comments` column specifically also has a lot of `Nan` values, so this column is good to drop for that reason as well.

We also want to drop the one missing value for Sex, so we recode the '.' to be a Nan value before dropping all the Nans from our data set to make training our model easier.

Lastly we recode the categorical variables to be numeric so that these columns will be easier to deal with when training and testing a model with these as features.

[5]:
```python
from sklearn import preprocessing

def clean_penguins_data(data):
    # make a copy of data frame in order to modify it
    df=data.copy()

    # drop useless columns
    df=df.drop(['studyName'],axis=1)
    df=df.drop(['Comments'],axis=1)
    df=df.drop(['Individual ID'],axis=1)
    df=df.drop(['Date Egg'],axis=1)
    df=df.drop(['Stage'],axis=1)

    # drop NAs (including '.')
    recode={
        "MALE":"M",
        "FEMALE":"F",
        ".":np.nan
    }
    df['Sex']=df['Sex'].map(recode)
    df=df.dropna()

    # recode categorical variables into numerical ones
    le = preprocessing.LabelEncoder()
    df['Sex'] = le.fit_transform(df['Sex'])
    df['Region'] = le.fit_transform(df['Region'])
    df['Island']= le.fit_transform(df['Island'])
    df['Clutch Completion']= le.fit_transform(df['Clutch Completion'])

    # predictor data X excluding Species column
    X=df.drop(['Species'],axis=1)

    # target data y has Species column
    y=df['Species']

    return(X,y)
```

[6]:
```python
X_train,y_train = clean_penguins_data(train)

X_test,y_test = clean_penguins_data(test)
```

[7]:
```python
X_train.shape, X_test.shape
```

```
[7]: ((260, 11), (64, 11))
```

3 Exploratory Analysis.

In this part, we will compute summary statistics and construct visualizations about the relationships between variables.

3.1 Displayed table

```
[27]: # suppress warnings in function
      import warnings
      warnings.filterwarnings('ignore')
```

```
[28]: # remove the one '.' data point for sex when looking at aggregate table.
      data = penguins[penguins['Sex'] != '.']

      data.groupby(["Species",'Sex'])[['Species',"Culmen Length (mm)",
                                        "Culmen Depth (mm)","Sex","Delta 15 N (o/
       ↪oo)",
                                        "Flipper Length (mm)","Body Mass (g)"]].
       ↪aggregate([len,np.mean,np.std]).round(2)
```

```
[28]:                                                    Culmen Length (mm)        \
                                                                      len   mean
      Species                                  Sex
      Adelie Penguin (Pygoscelis adeliae)      FEMALE                  73  37.26
                                               MALE                    73  40.39
      Chinstrap penguin (Pygoscelis antarctica) FEMALE                 34  46.57
                                               MALE                    34  51.09
      Gentoo penguin (Pygoscelis papua)        FEMALE                  58  45.56
                                               MALE                    61  49.47

                                                              Culmen Depth (mm)  \
                                                          std               len
      Species                                  Sex
      Adelie Penguin (Pygoscelis adeliae)      FEMALE  2.03                  73
                                               MALE    2.28                  73
      Chinstrap penguin (Pygoscelis antarctica) FEMALE  3.11                  34
                                               MALE    1.56                  34
      Gentoo penguin (Pygoscelis papua)        FEMALE  2.05                  58
                                               MALE    2.72                  61

                                                                         \
                                                         mean   std
      Species                                  Sex
      Adelie Penguin (Pygoscelis adeliae)      FEMALE  17.62  0.94
```

```
                                          MALE    19.07  1.02
Chinstrap penguin (Pygoscelis antarctica) FEMALE  17.59  0.78
                                          MALE    19.25  0.76
Gentoo penguin (Pygoscelis papua)         FEMALE  14.24  0.54
                                          MALE    15.72  0.74

                                                 Delta 15 N (o/oo)        \
                                                               len  mean
Species                                   Sex
Adelie Penguin (Pygoscelis adeliae)       FEMALE                73  8.79
                                          MALE                  73  8.93
Chinstrap penguin (Pygoscelis antarctica) FEMALE                34  9.25
                                          MALE                  34  9.46
Gentoo penguin (Pygoscelis papua)         FEMALE                58  8.19
                                          MALE                  61  8.30

                                                       Flipper Length (mm)  \
                                                   std                 len
Species                                   Sex
Adelie Penguin (Pygoscelis adeliae)       FEMALE  0.48                  73
                                          MALE    0.36                  73
Chinstrap penguin (Pygoscelis antarctica) FEMALE  0.32                  34
                                          MALE    0.39                  34
Gentoo penguin (Pygoscelis papua)         FEMALE  0.28                  58
                                          MALE    0.25                  61

                                                               Body Mass (g)  \
                                                    mean   std           len
Species                                   Sex
Adelie Penguin (Pygoscelis adeliae)       FEMALE  187.79  5.60            73
                                          MALE    192.41  6.60            73
Chinstrap penguin (Pygoscelis antarctica) FEMALE  191.74  5.75            34
                                          MALE    199.91  5.98            34
Gentoo penguin (Pygoscelis papua)         FEMALE  212.71  3.90            58
                                          MALE    221.54  5.67            61


                                                     mean     std
Species                                   Sex
Adelie Penguin (Pygoscelis adeliae)       FEMALE  3368.84  269.38
                                          MALE    4043.49  346.81
Chinstrap penguin (Pygoscelis antarctica) FEMALE  3527.21  285.33
                                          MALE    3938.97  362.14
Gentoo penguin (Pygoscelis papua)         FEMALE  4679.74  281.58
                                          MALE    5484.84  313.16
```

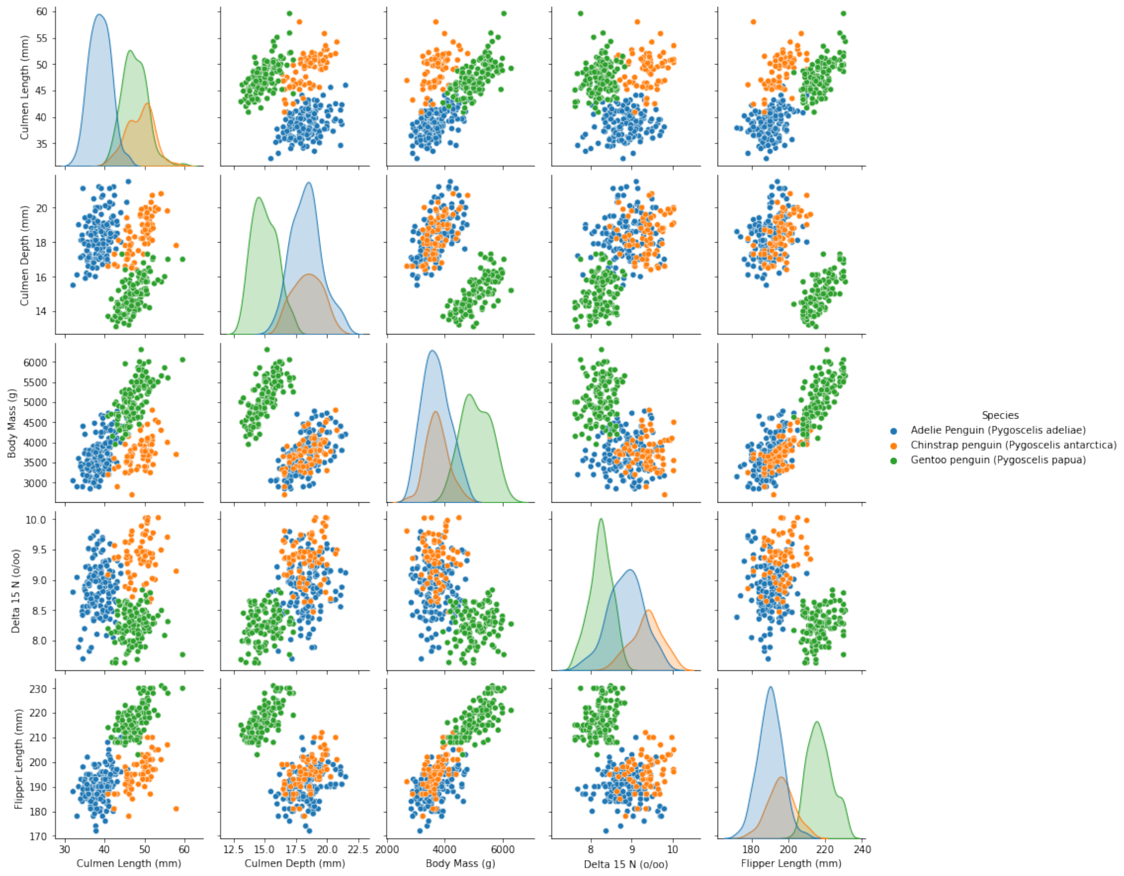
Looking at this table there are distinct differences in different species between culmen length, culmen depth with relatively small standard deviations.

There are also differences in Delta 15 N Body Mass and Flipper length, but these have larger standard deviations relative to these differences, so they may be more ambiguous between species.

3.2 Three Figures

3.2.1 Scatterplot

[9]:
```
import seaborn as sns
penguins_graph = penguins[['Species',"Culmen Length (mm)","Culmen Depth␣
↪(mm)","Sex",
                                "Island","Body Mass (g)","Delta 15 N (o/
↪oo)","Flipper Length (mm)"]]
sns.pairplot(penguins_graph, hue = 'Species')
plt.show()
```

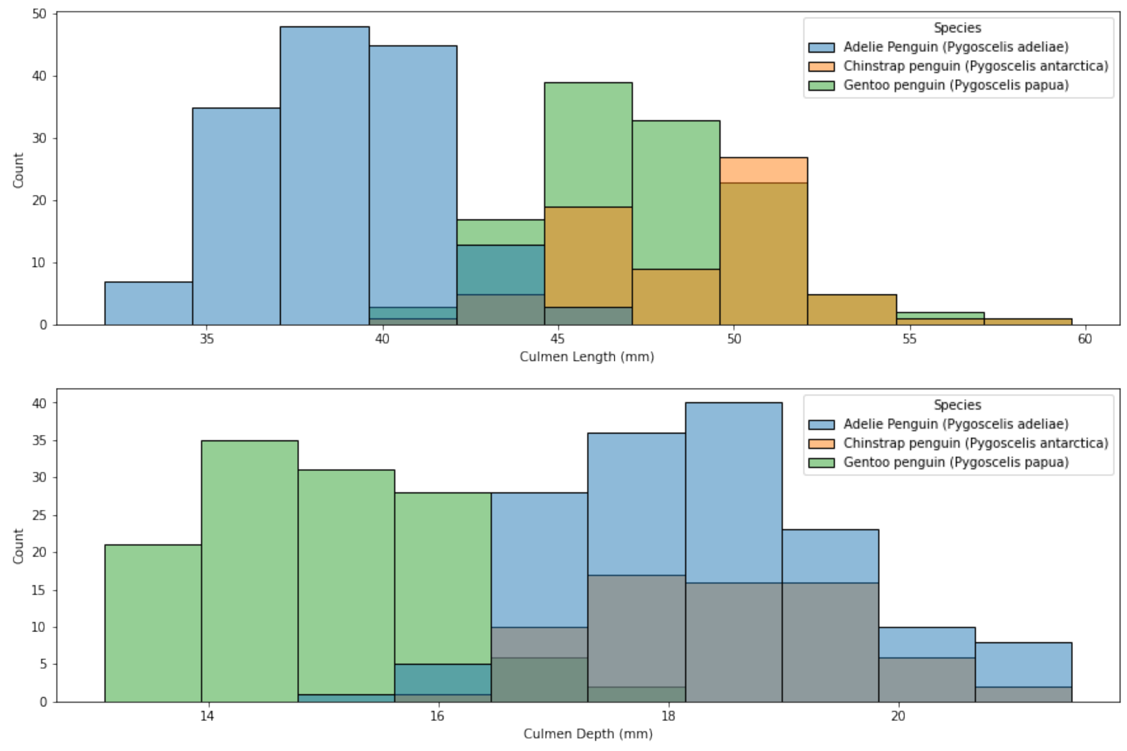


The scatterplot matrix gives us a little more insight into the distribution of the quantitative variables with respect to species and to each other. Culmen depth and body mass seems to be a good predictor for the gentoo penguin since in the scatterplot all the green gentoo data points are very clustered in a seperate region from the blue and yellow adelie and chinstrap penguin data points.

Both culmen length and culmen depth actually seem to be good predictors when paired with any variable, so these might be good to consider as features in a model.

3.2.2 Histogram

[10]:
```python
fig, ax = plt.subplots(2, 1, figsize = (15, 10))

sns.histplot(x = 'Culmen Length (mm)',
             hue = 'Species',
             data = penguins,
             ax = ax[0])

sns.histplot(x = 'Culmen Depth (mm)',
             hue = 'Species',
             data = penguins,
             ax = ax[1])

plt.show()
```



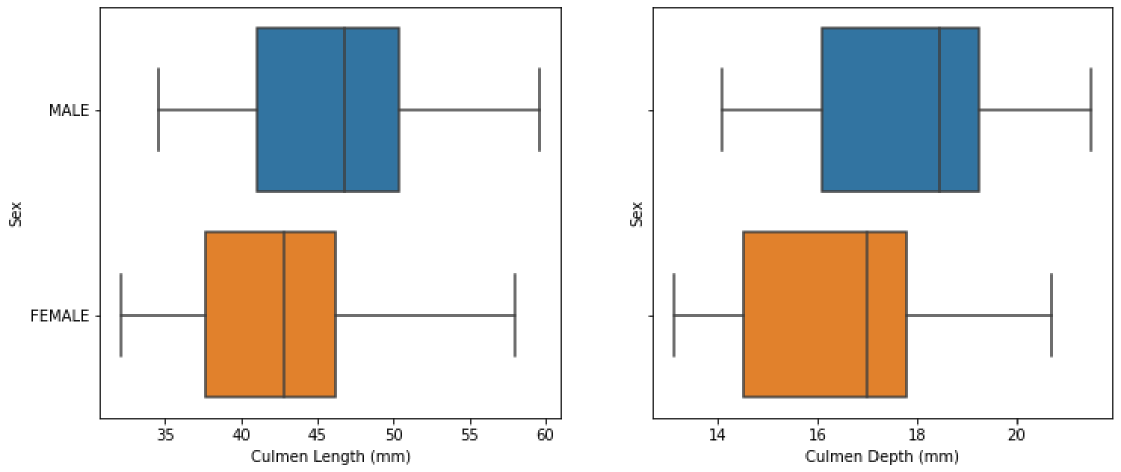
Looking at these histograms reveal some interesting patterns in the data. When we look at the distribution of culmen length and depth across species we can see some distinct groups within each variable. This is a good sign if we want to train a model using these features as predictors, as it will mean that knowing something about the culmen length and depth of a particular penguin will tell you a lot about what species that penguin is likely to be.

Moreover, these two features specifically seem like they will work well together because in terms of culmen length we can see two distinct groups—Adelie and Chinstrap/Gentoo—and in the culmen depth histogram we see two different groups—Gentoo and Adelie/Chinstrap.

Critically, knowing culmen length allows us to distinguish Adelie penguins from Chinstrap and Gentoo, and knowing culmen depth allows us to distinguish Gentoo from Chinstrap and Adelie, so using both culmen length and depth will tell us exactly what species that penguin is likely to be, which is perfect for our model.

3.2.3 Boxplot

[11]:
```
data = penguins[penguins['Sex'] != '.']

fig, ax = plt.subplots(1, 2, figsize = (12, 5), sharey = True)

sns.boxplot(x = 'Culmen Length (mm)',
            y = 'Sex',
            data = data,
            ax = ax[0])

sns.boxplot(x = 'Culmen Depth (mm)',
            y = 'Sex',
            data = data,
            ax = ax[1])

plt.show()
```



Based on this plot it might also make sense to use sex as a predictor for our model as well. If we are planning to use culmen length and depth as features for predicting, it may be a good idea to distinguish based on sex as well, since we can see clear differencees in the distribution of the culmen length and depth of penguins based on their sex. If we don't account for sex this could throw off our predictions and lead to some misclassification of penguins using only their culmen length and depth.

4 Feature Selection.

In this part, we will perform an analysis to choose one qualitative feature and two quantitative features, and justify our choice. We decide to test all combinations by our created function `comb_score`, which will train and evaluate a model by the Cross-validation score, and then return the best combination. The three models we are going to use for our feature selection are multinomial logistic regression, random forests, and support vector machines.

We will use cross validation in our feature selection over test score because we want to mitigate overfitting in our model, and cross validation will be better to make sure that doesn't happen.

[13]:
```
from sklearn.model_selection import cross_val_score
from itertools import combinations,permutations

# selected columns that we are going to evaluate
cols=["Culmen Length (mm)","Culmen Depth (mm)",
      "Delta 15 N (o/oo)","Flipper Length (mm)","Body Mass (g)"]

def comb_score(model,cols):
    """
    The function evaluates a model by Cross-validation
    model: machine learning models
    cols: selected columns being evaluate
    """

    # get all combinations
    comb_list = []
    qual_var = ["Sex", "Region", "Island"]
    combs = list(combinations(cols,2))
    for comb in combs:
        for var in qual_var:
            new_element = comb + (var,)
            comb_list.append(new_element)

    # create an list to store scores
    score = []

    # calculate every combinations' score
    for c in comb_list:
        s = cross_val_score(model,X_train[list(c)],y_train,cv = 5).mean()
        score.append(s)

    # print best score result
    print ("By " + str(model) + " model, the larget Cross-validation score is "+
           str(max(score))+ " and the selected combination is␣
 ↪"+str(comb_list[score.index(max(score))]))
```

We used a function called comb_scores to judge which combination would result in the highest model accuracy. In this model, we create a list of all possible permutations of the selected columns, after which we cross validated the scores using the given model. Then, it picks the highest of all combinations and prints the best result.

[14]:
```python
# find the best combination by using Logistic Regression model
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression()
comb_score(LR,cols)
```

```
By LogisticRegression() model, the larget Cross-validation score is
0.9884615384615385 and the selected combination is ('Culmen Length (mm)',
'Culmen Depth (mm)', 'Sex')
```

[15]:
```python
# find the best combination by using Random Forest model
from sklearn.ensemble import RandomForestClassifier
RFC = RandomForestClassifier()
comb_score(RFC,cols)
```

```
By RandomForestClassifier() model, the larget Cross-validation score is
0.9846153846153844 and the selected combination is ('Culmen Length (mm)', 'Body
Mass (g)', 'Sex')
```

[16]:
```python
# find the best combination by using Support Vector Machine model
from sklearn import svm
SVM = svm.SVC()
comb_score(SVM,cols)
```

```
By SVC() model, the larget Cross-validation score is 0.8384615384615385 and the
selected combination is ('Culmen Length (mm)', 'Culmen Depth (mm)', 'Island')
```

From the above results, we can see that both logistic regression and random forests model selected Culmen Length and Sex as variables used to determine the species. However, Logistic Regression slightly outperformed random forest. Thus, we conclude that the three most appropriate variables are 'Culmen Length (mm)', 'Culmen Depth (mm)', and 'Sex'. 'Sex' is the qualitative feature we selected and 'Culmen Length (mm)', 'Culmen Depth (mm)' are two quantitative features we selected.

5 Modeling

In this part, we will deploy at least three machine learning models and evaluate their performance. The three machine learning models are multinomial logistic regression, random forests, and support vector machines.

Start by defining a `plot_regions` function to visualize how our models are performing.

[17]:
```python
import matplotlib.patches as mpatches

def plot_regions(c,X,y, xlabel0, xlabel1, ylabel):
    """
    This function visualizes the decision regions for classifiers against data
    c: classifer
    X & y : predictor data & target data
    """

    x0 = X["Culmen Length (mm)"]
    x1 = X["Culmen Depth (mm)"]

    grid_x = np.linspace(x0.min(),x0.max(),501)
    grid_y = np.linspace(x1.min(),x1.max(),501)

    xx,yy = np.meshgrid(grid_x,grid_y)

    #male
    zz_m = np.zeros(xx.shape)
    #female
    zz_f = np.ones(xx.shape)

    XX = xx.ravel()
    YY = yy.ravel()
    ZZ_m = zz_m.ravel()
    ZZ_f = zz_f.ravel()

    p_m=c.predict(np.c_[XX,YY,ZZ_m])
    p_f=c.predict(np.c_[XX,YY,ZZ_f])

    p_m=p_m.reshape(xx.shape)
    p_f=p_f.reshape(xx.shape)

    # create plots
    fig,ax=plt.subplots(1, 2, sharey=True)

    #plot the decision regions
    ax[0].contourf(xx, yy, p_m, cmap="jet", alpha=.2)

    ax[0].scatter(x0, x1, c=y, cmap="jet")
    ax[0].set(xlabel=xlabel0, ylabel=ylabel)

    ax[1].contourf(xx, yy, p_f, cmap="jet", alpha=.2)

    ax[1].scatter(x0, x1, c=y, cmap="jet")
    ax[1].set(xlabel=xlabel1)

    blue_patch = mpatches.Patch(color='mediumblue', label='adelie')
    red_patch = mpatches.Patch(color='firebrick', label='chinstrap')
    green_patch = mpatches.Patch(color='limegreen', label='gentoo')
    plt.legend(handles=[blue_patch, red_patch, green_patch],
               bbox_to_anchor = (1.05, 1))
```

5.1 (1) Multinomial logistic regression.

Create the features (X) and the labels (y) and define a function to test the cross validation scores given the training data for MLR.

[18]:
```
cols = ["Culmen Length (mm)", "Culmen Depth (mm)", "Sex"]
X = X_train[cols]
y = y_train

def cv_score(cols):
    LR = LogisticRegression()
    print("Columns: " + str(cols))
    return cross_val_score(LR, X, y, cv = 5).mean()

cv_score(cols)
```

```
Columns: ['Culmen Length (mm)', 'Culmen Depth (mm)', 'Sex']
```

[18]:
```
0.9884615384615385
```

5.1.1 Model and Confusion Matrix

Create the MLR model using sklearn. Then define the confusion matrix to see how the model did.

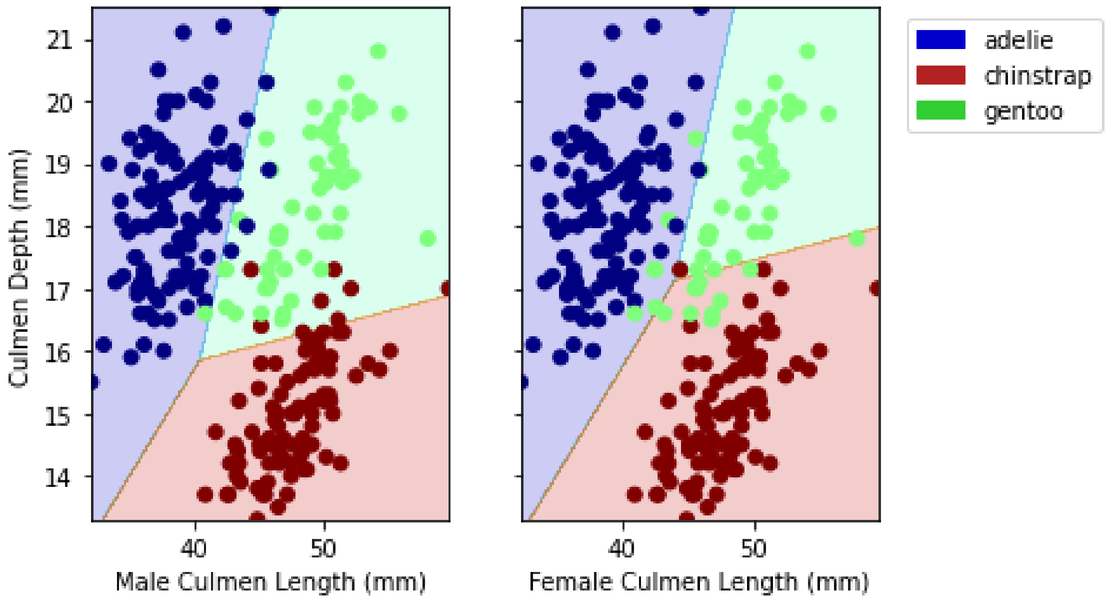
[19]:
```
from sklearn.metrics import confusion_matrix

LR = LogisticRegression()

# recode labels
le = preprocessing.LabelEncoder()

# fit models
X_test = X_test[cols]
y = le.fit_transform(y_train)
LR.fit(X,y)
y_pred = LR.predict(X_test)
y_test = le.fit_transform(y_test)

# create confusion matrix
confusion_matrix = confusion_matrix(y_test,y_pred)
confusion_matrix
```

```
[19]: array([[27,  0,  0],
             [ 0, 14,  0],
             [ 0,  0, 23]])
```

The confusion matrix shows that the MLR model accurately predicts species for all of the penguins in the test set.

5.1.2 Decision regions

```
[20]: # plot decision regions
      plot_regions(LR,X,y, "Male Culmen Length (mm)", "Female Culmen Length (mm)",␣
       ↪"Culmen Depth (mm)")
```



We use the function we defined earlier to visualize how the model is predicting species based on the features of the model.

We can see that the model performs pretty well on most of the points in the data set. Most points are in their corresponding region, meaning that the model classified them accurately.

There are a few outliers of each color that seem to have been mislabeled, but in the testing data all points were classified correctly.

5.2 (2) Random forests

Create the features (X) and the labels (y) and define a function to test the cross validation scores given the training data for random forest.

```
[21]: cols = ["Culmen Length (mm)", "Culmen Depth (mm)", "Sex"]
      X = X_train[cols]
```

```
y = y_train

def cv_score(cols):
    RFC = RandomForestClassifier()
    print("Columns: " + str(cols))
    return cross_val_score(RFC, X, y, cv = 5).mean()

cv_score(cols)
```

```
Columns: ['Culmen Length (mm)', 'Culmen Depth (mm)', 'Sex']
```

[21]:
```
0.9807692307692306
```

5.2.1 Model and Confusion Matrix

Create the random forest model using sklearn. Again, create the confusion matrix to see how the model did.

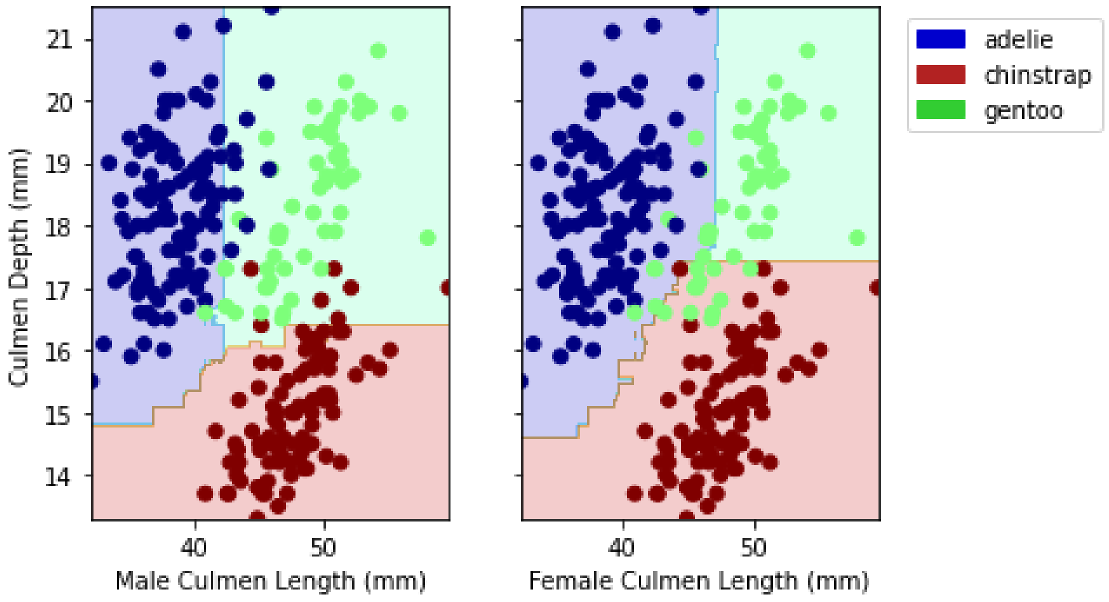
[22]:
```
from sklearn.metrics import confusion_matrix

RFC = RandomForestClassifier()

# recode labels
le = preprocessing.LabelEncoder()

# fit models
X_test = X_test[cols]
y = le.fit_transform(y_train)
RFC.fit(X,y)
y_pred = RFC.predict(X_test)
y_test = le.fit_transform(y_test)

# create confusion matrix
confusion_matrix = confusion_matrix(y_test,y_pred)
confusion_matrix
```

[22]:
```
array([[27,  0,  0],
       [ 0, 13,  1],
       [ 0,  0, 23]])
```

The confusion matrix shows that the random forest model also accurately predicted the species of 63/64 penguins in the test set. But it did confuse a gentoo penguin for a chinstrap penguin, which may have been an outlier.

5.2.2 Decision regions

[23]:
```
# plot decision regions
plot_regions(RFC,X,y, 'Male Culmen Length (mm)', 'Female Culmen Length (mm)',␣
 ↪'Culmen Depth (mm)')
```



Once again we can visualize how the model divided the data into species with the `plot_regions` function. This regions plot looks more complex than the MLR one, which makes sense for a random forest model.

There do seem to be some points that are missclassified, specifically in the female gentoo penguins being labeled as chinstrap by the model. This is likely what happened with the gentoo penguin that was misslabeled as chinstrap in the confusion matrix for the testing data above.

5.3 (3) Support vector machines

Create the features (X) and the labels (y) and define a function to test the cross validation scores given the training data for a support vector machines model.

[24]:
```
cols = ['Culmen Length (mm)', 'Culmen Depth (mm)', 'Sex']
X=X_train[cols]
y=le.fit_transform(y_train)

def cv_score(cols):
    SVM=svm.SVC()
    print("Columns: " + str(cols))
    return cross_val_score(SVM, X, y, cv=5).mean()
```

```
cv_score(cols)
```

```
Columns: ['Culmen Length (mm)', 'Culmen Depth (mm)', 'Sex']
```

[24]:
```
0.8269230769230769
```

Interestingly, this cross validation score is significantly lower than the first 2 models we trained.

5.3.1 Model and Confusion Matrix

Create the SVM model in sklearn and confusion matrix to evaluate it.

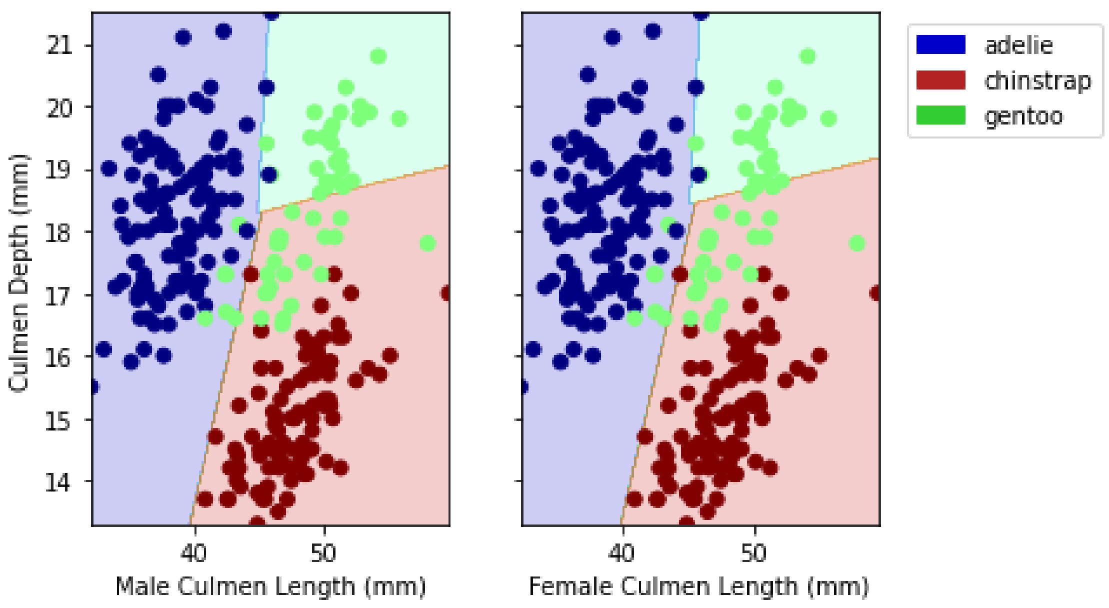
[25]:
```python
from sklearn.metrics import confusion_matrix

SVM=svm.SVC()

# recode labels
le = preprocessing.LabelEncoder()

# fit models
X_test = X_test[cols]
y = le.fit_transform(y_train)
SVM.fit(X,y)
y_pred = SVM.predict(X_test)
y_test = le.fit_transform(y_test)

# create confusion matrix
confusion_matrix = confusion_matrix(y_test,y_pred)
confusion_matrix
```

[25]:
```
array([[27,  0,  0],
       [ 0,  7,  7],
       [ 0,  0, 23]])
```

In the confusion matrix we see that the model predicted chinstrap for 7 penguins that were actually gentoo. So this model performed significantly worse on the data than the first 2. Let's look at the decision regions to see why.

5.3.2 Decision regions

[26]:
```python
# plot decision regions
plot_regions(SVM, X, y, 'Male Culmen Length (mm)', 'Female Culmen Length (mm)',␣
 ↪'Culmen Depth (mm)')
```



Visualizing the regions we can see that this support vector machine model is significantly less accurate than the first 2, which lines up with what we see in the scoring of the model.

We see that in many cases the model is confusing gentoo penguins for chinstrap penguins, as visualized by the large overlap of red chinstrap prediction region under the multitude of green gentoo data points.

This is happening the model is using culmen depth as a predictor of species, and as we can see from the decision regions, it classifies penguins of culmen depth less than ~ 18-19 mm to be chinstrap penguins. However, we know from looking at the real data, there are some gentoo penguins that have a culmen depth of less than ~ 18-19 mm, and so these gentoo penguins are misclassified by the model as chinstrap.

6 Discussion

6.0.1 Multiple Linear Regression

For the Logistic Regression model, the Cross-validation score is 0.988 that is very high. It means that the model is acurately predicting the species of the penguin when it is provided with the penguin's Culmen Length, Culmen Depth, and Sex.

By the confusion matrix, the model correctly classified all the penguins' species of the test data. When looking at the decision regions, we see that this model performs very well. There are a few points which lie outside of the regions, but many of the points incorrectly predicted are outliers and lie closer to other species than their own.

With regards to overfitting, we don't think that the MLR model fell victim to overfitting. The model makes some mistakes in the cross validation, but overall seems to do about as well on the testing data as it does on the cv and training data. And based on the decision regions plot, it looks like the model will be able to generalize well to more data beyond what we have in this dataset.

To improve this model we could get more data, and then select more features as predictors for our model.

6.0.2 Random Forest

For the Random Forest model, the Cross-validation score is 0.981 that is also high. It means that the model is acurately predicting the species of the penguin when it is provided with the penguin's Culmen Length, Culmen Depth, and Sex.

By the confusion matrix, 1 of the 64 penguins from the test data is predictied incorrectly. In general, the model performs well. If we want to improve the model's performance, we could find a good combination by GridSearch or Bayesian optimization.

The random forest model also does not seem to be guilty of overfitting to the training data. Again, this model does about equally well on the testing data as it does the training data. The decision regions plot may look a little bit wonky, but there's no evidence that the regions are overly conforming to the training data. This model would likely be able to generalize well to new data.

6.0.3 Support Vector Machines

For the Support Vector Machines model, the Cross-validation score is 0.827. That is lower than the previous two models. It means that the model will acurately predict most of species of the penguin when it is provided with the penguin's Culmen Length, Culmen Depth, and Sex.

By the confusion matrix, 7 of the 64 penguins from the test data is predictied incorrectly. When doing feature selection, initially the code ran for all possible combinations of columns. This model performed significantly better when it was using three qualitative variables to predict penguin species, so if we expanded the amount of variables we could use to include more qualitative variables, The Support Vector Machines model would likely perform significantly better than it did. We can see some of this error in the decision trees, where a large clump of the gentoo species was predicted as chinstrap.

Considering overfitting, this model seems to be fine as well. It scores about as well on the testing data as it does on the training data. Looking at the decision regions plot this model also doesn't show any visible signs of overfitting. It will likely be able to generalize well to new data.

Overall, all our models seemed to perform very well. The multiple logisitc regression and random forest models both predicted the species for 100% of the testing data. So either of these models would be valid.

The support vector machine model performed decidedly worse than the other two so we wouldn't recommend it.

In terms of choosing a final model that we reccomend, we would say the MLR model is better than the random forest for 2 reasons. - 1. The MLR model had a very slightly better cross-validation score than the random forest. - 2. The MLR model is much simpler than the random forest, meaning that it is physically smaller in terms of object size, and it will likely generalize better to more data since it is simpler.

So to reiterate, the final model we chose was a multiple linear regression model that used `Culmen Length (mm)`, `Culmen depth (mm)` and `Sex` as predictors.