Penguins_Group_Project

March 9, 2022

0.1 PIC 16A: Penguins Group Project

Group Members: Jim Liu

Yijun Fang

Morgan Chan

0.2 1. Group Contribution Statement

All three of us wrote the data acquisition, preparation and feature selection, and each contributed a figure. Jim worked on 1,2 (scatter plot and histogram) and wrote the explanation for data cleaning. Morgan worked on the barplot and wrote the explanation for exploratory tables and figures. Yijun worked on the pairplots and wrote the explanation for feature selection.For the machine learning models, Morgan led the logistic regression model. Jim led the random forest model. Yijun led the K-Neighbors model. We each wrote the explanations for each figure and each model we did, and each member worked on the conclusion collaboratively. We all checked each other's work and made revisions to code and writing.

0.3 2. Data Import and Cleaning

2.1 Importing libraries and penguins dataset

[206]:
```
# import libraries
import pandas as pd
from matplotlib import pyplot as plt
from sklearn import tree, preprocessing
from sklearn.model_selection import train_test_split
import numpy as np
import seaborn as sns

import os, ssl
if (not os.environ.get('PYTHONHTTPSVERIFY', '') and getattr(ssl,␣
 ↪'_create_unverified_context', None)):
    ssl._create_default_https_context = ssl._create_unverified_context

# import data
url = "https://philchodrow.github.io/PIC16A/datasets/palmer_penguins.csv"
penguins = pd.read_csv(url)
penguins
```

[206]:

	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	
..	…	…	…	…	
339	PAL0910	120	Gentoo penguin (Pygoscelis papua)	Anvers	
340	PAL0910	121	Gentoo penguin (Pygoscelis papua)	Anvers	
341	PAL0910	122	Gentoo penguin (Pygoscelis papua)	Anvers	
342	PAL0910	123	Gentoo penguin (Pygoscelis papua)	Anvers	
343	PAL0910	124	Gentoo penguin (Pygoscelis papua)	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	
..	…	…	…	…	…	
339	Biscoe	Adult, 1 Egg Stage	N38A2	No	12/1/09	
340	Biscoe	Adult, 1 Egg Stage	N39A1	Yes	11/22/09	
341	Biscoe	Adult, 1 Egg Stage	N39A2	Yes	11/22/09	
342	Biscoe	Adult, 1 Egg Stage	N43A1	Yes	11/22/09	
343	Biscoe	Adult, 1 Egg Stage	N43A2	Yes	11/22/09	

	Culmen Length (mm)	Culmen Depth (mm)	Flipper Length (mm)	\
0	39.1	18.7	181.0	
1	39.5	17.4	186.0	
2	40.3	18.0	195.0	
3	NaN	NaN	NaN	
4	36.7	19.3	193.0	
..	…	…	…	
339	NaN	NaN	NaN	
340	46.8	14.3	215.0	
341	50.4	15.7	222.0	
342	45.2	14.8	212.0	
343	49.9	16.1	213.0	

	Body Mass (g)	Sex	Delta 15 N (o/oo)	Delta 13 C (o/oo)	\
0	3750.0	MALE	NaN	NaN	
1	3800.0	FEMALE	8.94956	-24.69454	
2	3250.0	FEMALE	8.36821	-25.33302	
3	NaN	NaN	NaN	NaN	
4	3450.0	FEMALE	8.76651	-25.32426	
..	…	…	…	…	
339	NaN	NaN	NaN	NaN	

```
340         4850.0  FEMALE            8.41151          -26.13832
341         5750.0    MALE            8.30166          -26.04117
342         5200.0  FEMALE            8.24246          -26.11969
343         5400.0    MALE            8.36390          -26.15531

                            Comments
0    Not enough blood for isotopes.
1                                NaN
2                                NaN
3                Adult not sampled.
4                                NaN
..                               …
339                              NaN
340                              NaN
341                              NaN
342                              NaN
343                              NaN

[344 rows x 17 columns]
```

[207]:
```
# drop unwanted columns
drop_cols=["studyName","Sample Number","Region","Clutch␣
 ↪Completion","Stage","Individual ID","Date Egg","Comments"]
penguins=penguins.drop(labels=drop_cols,axis=1)

# change the "." in "Sex" column to NAN and remove rows with NAN
penguins['Sex'][penguins["Sex"]=="."]=np.nan
penguins = penguins.dropna()
penguins
```

```
/var/folders/90/p48m2gv52q9g0xmtw8_zqqtm0000gn/T/ipykernel_8448/445936563.py:6:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  penguins['Sex'][penguins["Sex"]=="."]=np.nan
```

[207]:
```
                                   Species     Island  Culmen Length (mm)  \
1    Adelie Penguin (Pygoscelis adeliae)  Torgersen                39.5
2    Adelie Penguin (Pygoscelis adeliae)  Torgersen                40.3
4    Adelie Penguin (Pygoscelis adeliae)  Torgersen                36.7
5    Adelie Penguin (Pygoscelis adeliae)  Torgersen                39.3
6    Adelie Penguin (Pygoscelis adeliae)  Torgersen                38.9
..                                   …          …                   …
338    Gentoo penguin (Pygoscelis papua)     Biscoe                47.2
340    Gentoo penguin (Pygoscelis papua)     Biscoe                46.8
341    Gentoo penguin (Pygoscelis papua)     Biscoe                50.4
```

```
342    Gentoo penguin (Pygoscelis papua)     Biscoe                45.2
343    Gentoo penguin (Pygoscelis papua)     Biscoe                49.9

     Culmen Depth (mm)  Flipper Length (mm)  Body Mass (g)     Sex  \
1                 17.4                186.0         3800.0  FEMALE
2                 18.0                195.0         3250.0  FEMALE
4                 19.3                193.0         3450.0  FEMALE
5                 20.6                190.0         3650.0    MALE
6                 17.8                181.0         3625.0  FEMALE
..                 …                  …             …      …
338               13.7                214.0         4925.0  FEMALE
340               14.3                215.0         4850.0  FEMALE
341               15.7                222.0         5750.0    MALE
342               14.8                212.0         5200.0  FEMALE
343               16.1                213.0         5400.0    MALE

     Delta 15 N (o/oo)  Delta 13 C (o/oo)
1              8.94956          -24.69454
2              8.36821          -25.33302
4              8.76651          -25.32426
5              8.66496          -25.29805
6              9.18718          -25.21799
..                 …                  …
338            7.99184          -26.20538
340            8.41151          -26.13832
341            8.30166          -26.04117
342            8.24246          -26.11969
343            8.36390          -26.15531

[324 rows x 9 columns]
```

2.2 Seperate data into training and testing sets

[208]:
```
# spliting into 80% training set and 20% testing set
train, test = train_test_split(penguins, test_size = 0.2)
```

2.3 Data Cleaning

[200]:
```
def prep_penguin_data (df):
    '''
    This function cleans and prepares the penguin dataset by:
    1. transform qualitative variable into integers
    2. split dataframe into predictor X and response y
    3. returns X and y
    parameter df is the input dataframe
    '''
    # copies data from input
    df = df.copy()
```

```
    # transform qualitative variable into integers
    le = preprocessing.LabelEncoder()
    df["Sex"] = le.fit_transform(df["Sex"])
    df["Species"] = le.fit_transform(df["Species"])
    df["Island"] = le.fit_transform(df["Island"])

    # split dataframe into predictor X and response y
    X = df.drop(["Species"], axis = 1)
    y = df["Species"]

    return (X,y)
```

[201]:
```
# Prepares X_train, y_train, X_test, y_test
X_train, y_train = prep_penguin_data(train)
X_test, y_test = prep_penguin_data(test)
```

0.4 3. Exploratory Analysis

3.1 Summary Statistics Table We create a summary statistics table with a function that outputs a table of means, medians, and standard deviations for the cleaned penguins dataframe so that we can conduct exploratory analysis on the data.

[37]:
```
# Creating summary statistics table for the cleaned up penguins dataframe
def summary_stats_table(qual_cols, quan_cols):
    """
    This function returns a table of summary statistics for the penguins␣
 ↪dataframe
    """
    return penguins.groupby(qual_cols)[quan_cols].aggregate([np.mean,np.std,np.
 ↪median])

penguins["Species"]=penguins["Species"].str.split().str.get(0)
```

[38]:
```
summary_stats_table(['Species','Island'],['Culmen Length (mm)','Culmen Depth␣
 ↪(mm)','Flipper Length (mm)','Body Mass (g)','Delta 15 N (o/oo)','Delta 13 C␣
 ↪(o/oo)'])
```

[38]:

		Culmen Length (mm)			Culmen Depth (mm) \
		mean	std	median	mean
Species	Island				
Adelie	Biscoe	38.975000	2.480916	38.70	18.370455
	Dream	38.401923	2.501175	38.20	18.205769
	Torgersen	39.055814	3.129681	39.00	18.416279
Chinstrap	Dream	48.788060	3.342904	49.50	18.404478
Gentoo	Biscoe	47.570339	3.119262	47.45	14.994068

		Flipper Length (mm)					\
		std	median		mean	std	median
Species	Island						
Adelie	Biscoe	1.188820	18.45		188.795455	6.729247	189.5
	Dream	1.161880	18.10		190.096154	6.481526	190.0
	Torgersen	1.332547	18.40		192.162791	6.011617	191.0
Chinstrap	Dream	1.136106	18.40		195.671642	7.074041	196.0
Gentoo	Biscoe	0.989802	15.00		217.194915	6.598703	216.0

		Body Mass (g)			Delta 15 N (o/oo)	\
		mean	std	median	mean	
Species	Island					
Adelie	Biscoe	3709.659091	487.733722	3750.0	8.823593	
	Dream	3684.615385	439.366574	3575.0	8.948276	
	Torgersen	3717.441860	465.971045	3650.0	8.788554	
Chinstrap	Dream	3729.850746	386.300411	3700.0	9.356155	
Gentoo	Biscoe	5091.101695	503.402158	5050.0	8.249349	

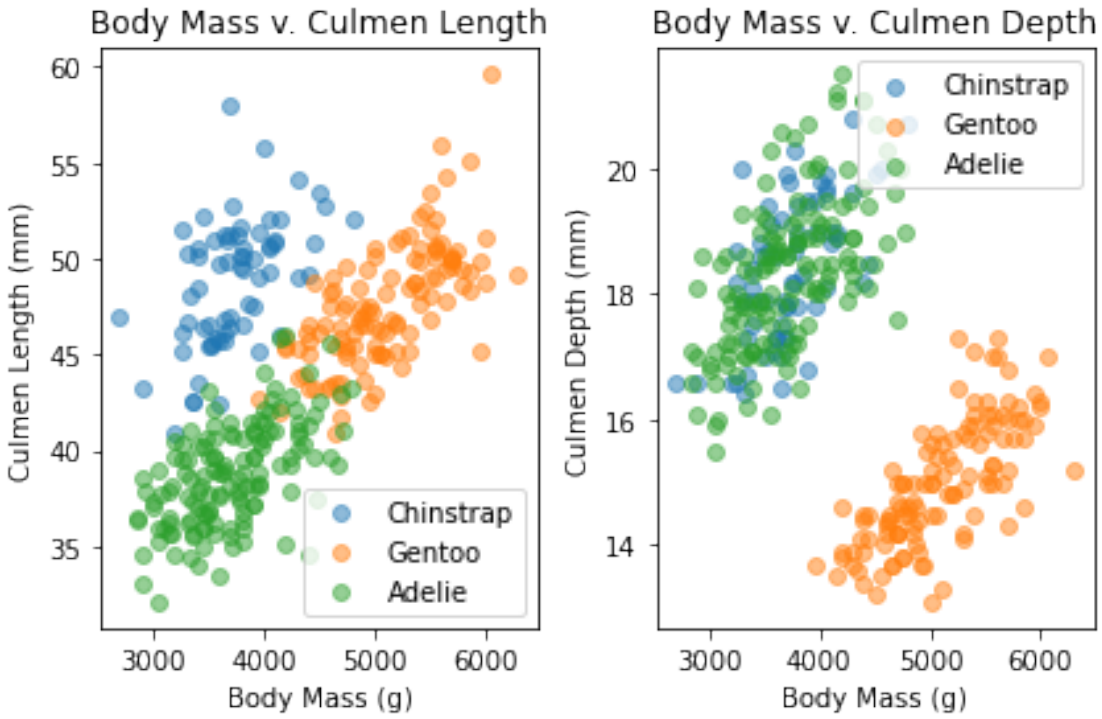
		Delta 13 C (o/oo)				
		std	median	mean	std	median
Species	Island					
Adelie	Biscoe	0.382470	8.787540	-25.918702	0.546280	-26.066955
	Dream	0.422908	8.965115	-25.747446	0.597462	-25.932470
	Torgersen	0.468310	8.868530	-25.785917	0.613047	-25.953990
Chinstrap	Dream	0.368720	9.373690	-24.557869	0.221445	-24.579940
Gentoo	Biscoe	0.267013	8.260440	-26.183681	0.542648	-26.221575

From the summary table, a piece of information we obtain is that Adelie occupies all three islands, whereas Chinstrap and Gentoo occupy one island each which helps us distinguish species by location. We will explore the numerical data in the table further in the following figures.

3.2 Figures Using the penguins data, we create figures to model the relationship between different variables and columns in the dataset.

[39]:
```
# Creating first figure

# Create empty subplots
fig,ax = plt.subplots(1,2)

# Define data and add it to subplots as scatter points
for i in set(penguins['Species']):
    s = penguins[penguins['Species'] == i]
    ax[0].scatter(s['Body Mass (g)'],
         s['Culmen Length (mm)'],
         label = i.split(' ')[0],
         alpha = 0.5)
    ax[1].scatter(s['Body Mass (g)'],
```

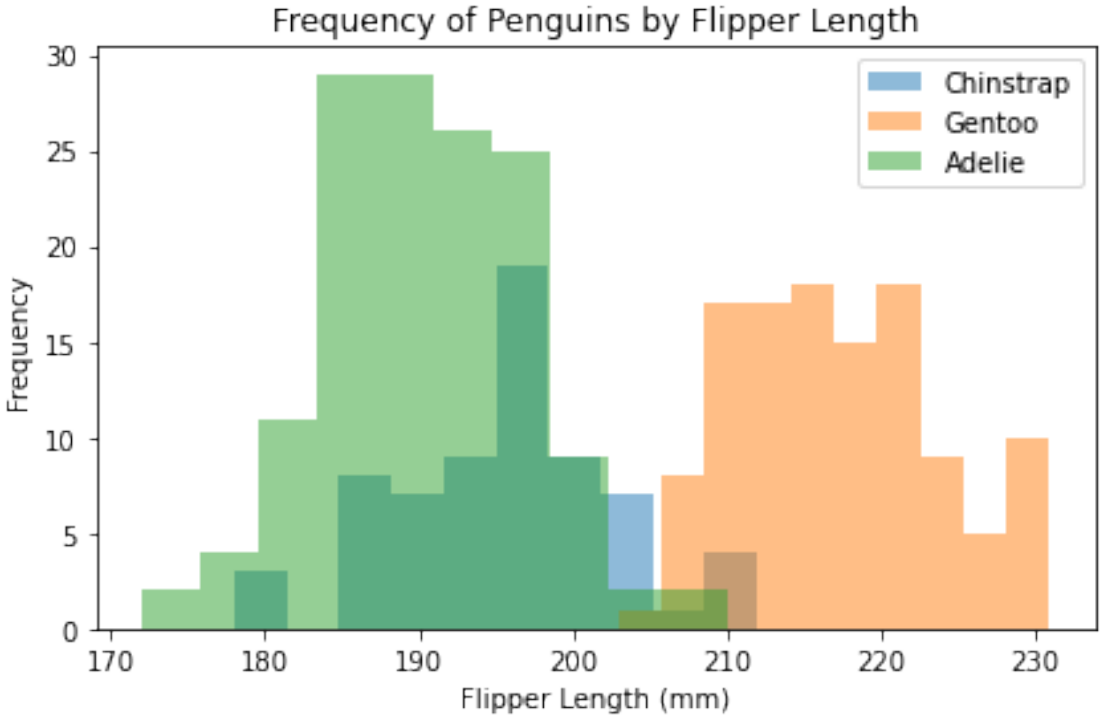
```
        s['Culmen Depth (mm)'],
        label = i.split(' ')[0],
        alpha = 0.5)

# Add labels and legend to plots
ax[0].set(title = "Body Mass v. Culmen Length",
          xlabel = "Body Mass (g)",
          ylabel = "Culmen Length (mm)")
ax[1].set(title = "Body Mass v. Culmen Depth",
          xlabel = "Body Mass (g)",
          ylabel = "Culmen Depth (mm)")
ax[0].legend()
ax[1].legend()
plt.tight_layout()
```



From the first figure we created, we plotted two scatterplots of Body Mass versus Culmen Length and Body Mass versus Culmen Depth. The plots suggest a positive correlation between both relationships in all three species, so a larger body mass correlates to larger culmen length and larger culmen depth. We can use this analysis to determine the range of correlation in both plots to distinguish the each penguin species.

[40]:
```
# Creating second figure

# Create empty subplot
```

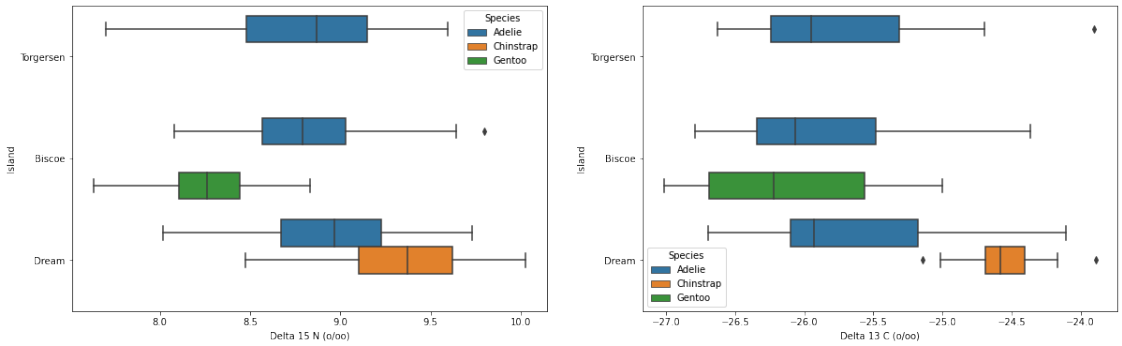
```
fig,ax = plt.subplots(1)

# Define data and add it to subplot as histogram
for i in set(penguins['Species']):
    s = penguins[penguins['Species'] == i]
    ax.hist(s['Flipper Length (mm)'],
            label = i.split(' ')[0],
            alpha = 0.5)

# Add labels and legend to plots
ax.set(title = "Frequency of Penguins by Flipper Length",
          xlabel = "Flipper Length (mm)",
          ylabel = "Frequency")
ax.legend()
plt.tight_layout()
```



From the second figure we created, we plotted a histogram of the frequency of penguins by flipper length. The plot which allows us to compare flipper lengths of different penguins suggests that Gentoos have larger flipper lengths than Adelies and Chinstraps. We can use this analysis to determine the average range of flipper length for each penguin species.

[41]:
```
# Creating third figure

# Create empty subplots
```

```
fig,(ax1,ax2) = plt.subplots(1,2, figsize=(20,6))

# Add data to plots as boxplots
fgrid = sns.boxplot(data = penguins,
                    x = "Delta 15 N (o/oo)",
                    y = "Island",
                    hue = "Species", ax = ax1)
fgrid = sns.boxplot(data = penguins,
                    x = "Delta 13 C (o/oo)",
                    y = "Island",
                    hue = "Species", ax = ax2)

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

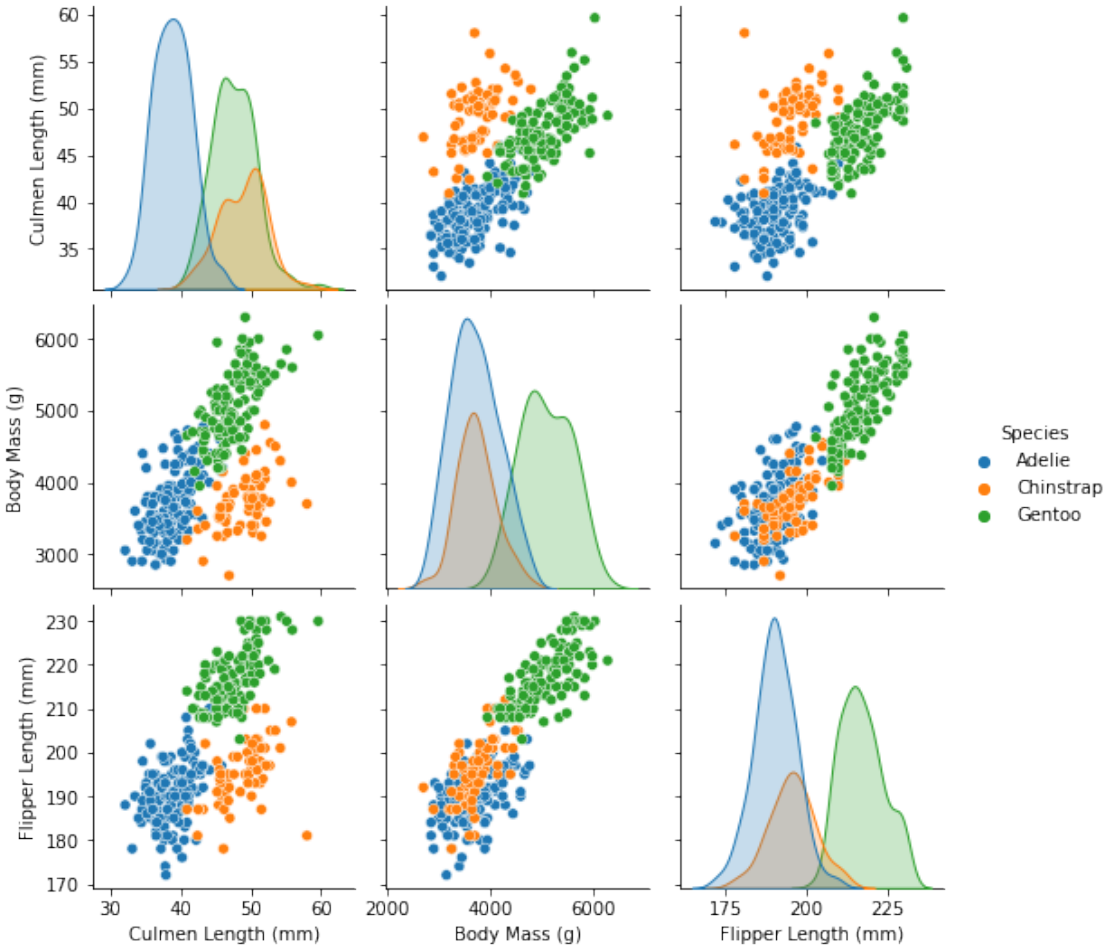



```
<Figure size 432x288 with 0 Axes>
```

From the third figure we created, we plotted two boxplots of Delta 15 N (o/oo) by island and species and Delta 13 N (o/oo) by island and species. The plots suggest that the pebguins tend to have higher Delta 15 N than Delta 13 N and that the Chinstraps tend to have higher Delta 15 N and Delta 13 N than the other penguins whereas Gentoos tend to have smaller Delta 15 and 13 N than the other penguins. We can use this analysis to distinguish the average Delta 15 N and Delta 13 N for each penguin species.

[42]:
```
cols = ["Species", "Culmen Length (mm)", "Body Mass (g)", "Flipper Length (mm)"]
sns.pairplot(penguins[cols], hue = "Species")
```

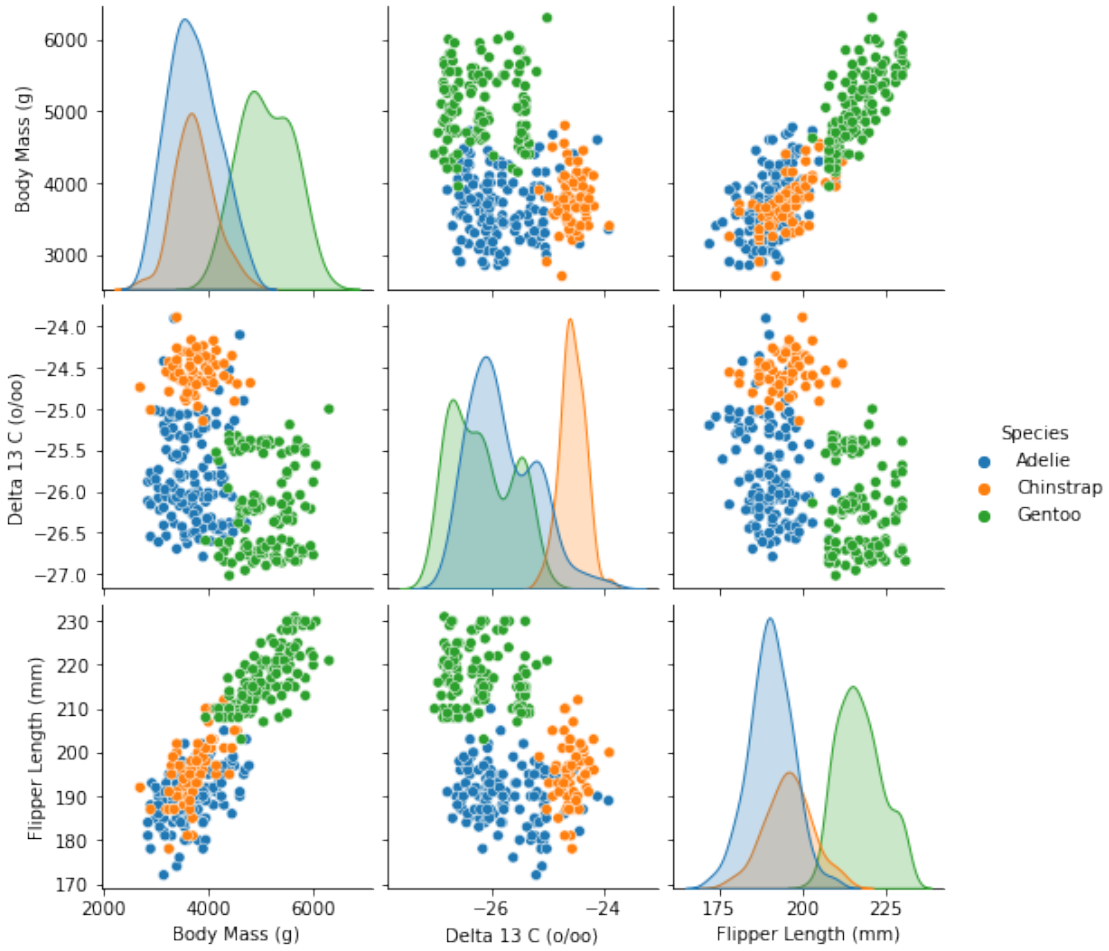
[42]:
```
<seaborn.axisgrid.PairGrid at 0x7fc0a2909d90>
```



The figures suggest that there is a similar range for the Adelie and Chinstrap in body mass versus flipper length, which helps us exclude the features that are less distinguishable in our models. The figures also suggest that there are differing ranges for all three species for culmen length versus body mass, which allows us to further test these features in the next section.

[43]:
```
cols = ["Species", "Body Mass (g)", "Delta 13 C (o/oo)", "Flipper Length (mm)"]
sns.pairplot(penguins[cols], hue = "Species")
```

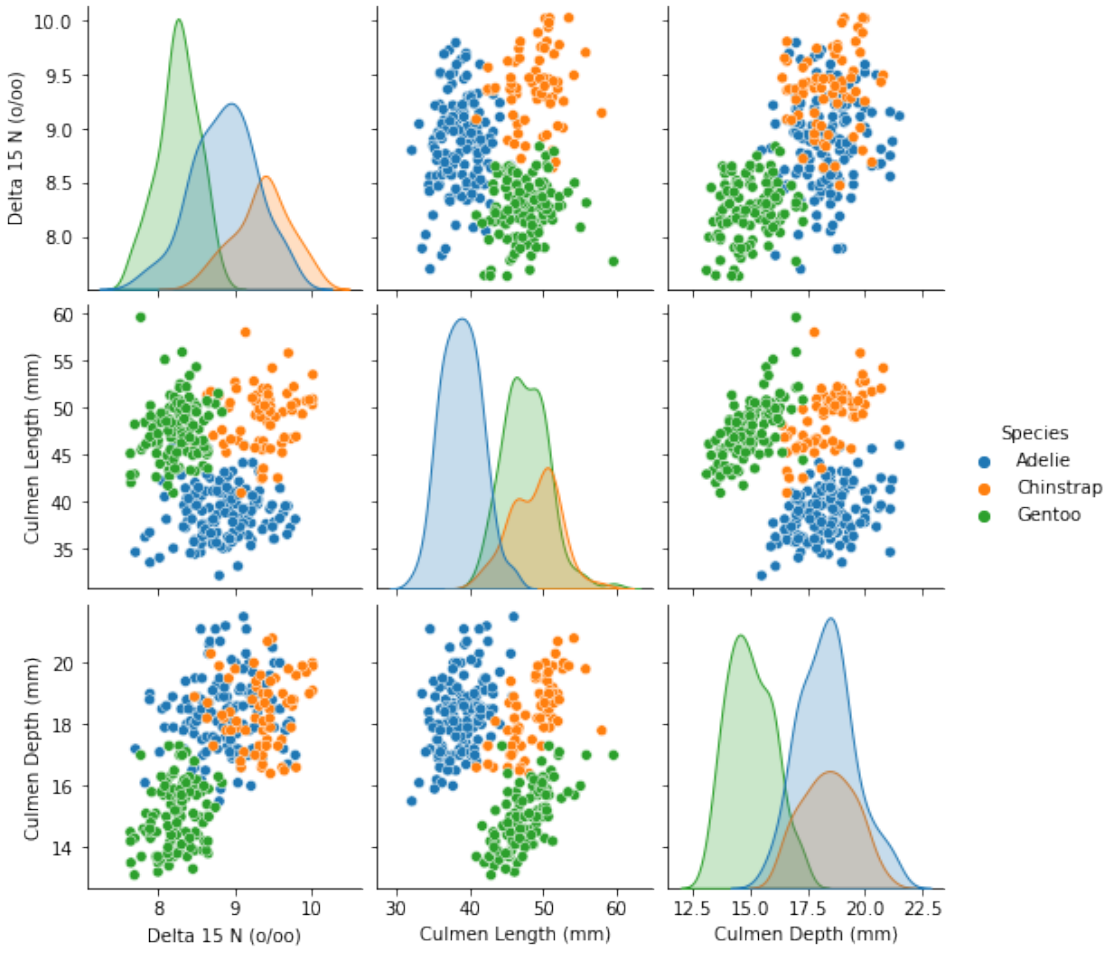
[43]: <seaborn.axisgrid.PairGrid at 0x7fc0a342d400>



The figures suggest that there is a similar range for the Adelie and Chinstrap in body mass versus flipper length, which helps us exclude the features that are less distinguishable in our models. The figures also suggest that there are differing ranges for all three species for Delta 13 C versus flipper length which allows us to further test these features in the next section.

[44]:
```
cols = ["Species", "Delta 15 N (o/oo)", "Culmen Length (mm)", "Culmen Depth␣
↪(mm)"]
sns.pairplot(penguins[cols], hue = "Species")
```

[44]: <seaborn.axisgrid.PairGrid at 0x7fc0a342d190>



The figures suggest that there is a similar range for the Adelie and Chinstrap in Delta 15 N and in culmen length, which helps us exclude the features that are less distinguishable in our models. The figures also suggest that there are differing ranges for all three species for culmen depth and culmen length which allows us to further test these features in the next section.

0.5 4. Feature Selection

In part four feature selection, we are going to select predictor variables for our three machine learning models. These predictor variable will include one qualitative variable and two quantitative variables. Feature selection will help the models get rid of the redundant features of the data set and perform better. The models we will use in part five are Logistic Regression, Random Forest and K-Neighbor. The function cal_feature_score will print the cross validation score of these models with given features.

[210]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
```

```
#set max iteration number to 1000 for LogisticRegression model to avoid␣
 ↪unlimited iterations.
LR = LogisticRegression(max_iter = 1000)
#set parameters to default for the K-Neighbors and RandomForest models
KN = KNeighborsClassifier()
RF = RandomForestClassifier()

def cal_feature_score(model,cols):
    """
    This function calculates the cross validation score on the
    features of the data
    """

    print("training with columns" + str(cols))
    return cross_val_score(model,X_train[cols],y_train,cv=5).mean()
```

Then, we will select some possible combinations of features. In the pair correlation graphs in part 3, some quantitative predictor pairs give a clearer separation across different species compared to others. We suppose that these combinations will give a higher cross-validation scores. We will loop over all of these combinations and select the one with highest score for each model.

[211]:
```
combos = [['Sex', 'Body Mass (g)', 'Culmen Length (mm)'],
          ['Island', 'Body Mass (g)', 'Culmen Length (mm)'],
          ["Island",'Flipper Length (mm)', 'Culmen Length (mm)'],
          ["Sex",'Flipper Length (mm)', 'Culmen Length (mm)'],
          ["Sex",'Flipper Length (mm)', 'Delta 13 C (o/oo)'],
          ["Island",'Flipper Length (mm)', 'Delta 13 C (o/oo)'],
          ["Sex",'Culmen Depth (mm)', 'Culmen Length (mm)'],
          ["Island",'Culmen Depth (mm)', 'Culmen Length (mm)']
          ]
```

CV scores of K-Neighbor Model

[212]:
```
for combo in combos:
    x=cal_feature_score(KN, combo)
    print("CV score is "+ str(np.round(x,3)))
```

```
training with columns['Sex', 'Body Mass (g)', 'Culmen Length (mm)']
CV score is 0.745
training with columns['Island', 'Body Mass (g)', 'Culmen Length (mm)']
CV score is 0.745
training with columns['Island', 'Flipper Length (mm)', 'Culmen Length (mm)']
CV score is 0.95
training with columns['Sex', 'Flipper Length (mm)', 'Culmen Length (mm)']
CV score is 0.95
training with columns['Sex', 'Flipper Length (mm)', 'Delta 13 C (o/oo)']
CV score is 0.873
training with columns['Island', 'Flipper Length (mm)', 'Delta 13 C (o/oo)']
```

```
CV score is 0.888
training with columns['Sex', 'Culmen Depth (mm)', 'Culmen Length (mm)']
CV score is 0.973
training with columns['Island', 'Culmen Depth (mm)', 'Culmen Length (mm)']
CV score is 0.969
```

CV scores of Logistic Regression Model

[213]:
```
for combo in combos:
    x=cal_feature_score(LR, combo)
    print("CV score is "+ str(np.round(x,3)))
```

```
training with columns['Sex', 'Body Mass (g)', 'Culmen Length (mm)']
CV score is 0.965
training with columns['Island', 'Body Mass (g)', 'Culmen Length (mm)']
CV score is 0.965
training with columns['Island', 'Flipper Length (mm)', 'Culmen Length (mm)']
CV score is 0.95
training with columns['Sex', 'Flipper Length (mm)', 'Culmen Length (mm)']
CV score is 0.95
training with columns['Sex', 'Flipper Length (mm)', 'Delta 13 C (o/oo)']
CV score is 0.942
training with columns['Island', 'Flipper Length (mm)', 'Delta 13 C (o/oo)']
CV score is 0.95
training with columns['Sex', 'Culmen Depth (mm)', 'Culmen Length (mm)']
CV score is 0.988
training with columns['Island', 'Culmen Depth (mm)', 'Culmen Length (mm)']
CV score is 0.973
```

CV scores of Random Forest Model

[214]:
```
for combo in combos:
    x=cal_feature_score(RF, combo)
    print("CV score is "+ str(np.round(x,3)))
```

```
training with columns['Sex', 'Body Mass (g)', 'Culmen Length (mm)']
CV score is 0.973
training with columns['Island', 'Body Mass (g)', 'Culmen Length (mm)']
CV score is 0.946
training with columns['Island', 'Flipper Length (mm)', 'Culmen Length (mm)']
CV score is 0.965
training with columns['Sex', 'Flipper Length (mm)', 'Culmen Length (mm)']
CV score is 0.981
training with columns['Sex', 'Flipper Length (mm)', 'Delta 13 C (o/oo)']
CV score is 0.923
training with columns['Island', 'Flipper Length (mm)', 'Delta 13 C (o/oo)']
CV score is 0.965
training with columns['Sex', 'Culmen Depth (mm)', 'Culmen Length (mm)']
CV score is 0.985
training with columns['Island', 'Culmen Depth (mm)', 'Culmen Length (mm)']
```

```
CV score is 0.973
```

From the above results, we notice that for K-Neighbor model, the most effective feature combination is ['Sex', 'Culmen Depth (mm)', 'Culmen Length (mm)'], while for the other two model, it is ['Island', 'Culmen Depth (mm)', 'Culmen Length (mm)']. In part five, we will train and test our models based on this result.

A brief note that the current output shows ['Sex', 'Culmen Depth (mm)', 'Culmen Length (mm)'] has a higher CV score than ['Island', 'Culmen Depth (mm)', 'Culmen Length (mm)'] for logistic regression and random forest models; however, the first time we ran the code, ['Island', 'Culmen Depth (mm)', 'Culmen Length (mm)'] had a higher CV score, so we based our logistic regression and random forest models on it, and we accidentally reran the code when reviewing the project.

0.6 5. Modeling

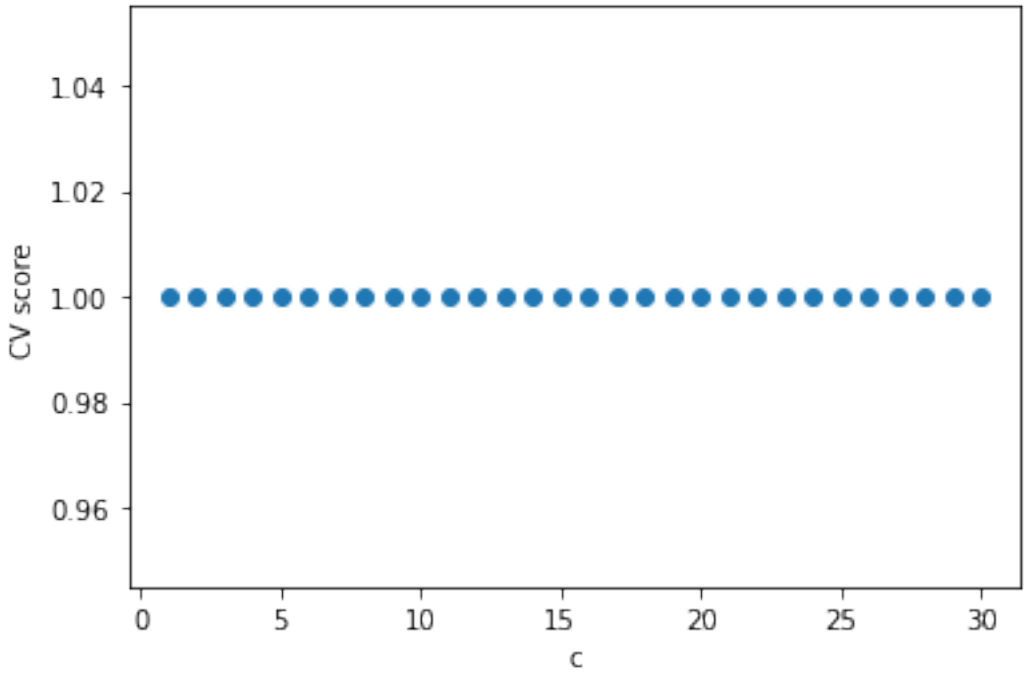
5.1 Logistic Regression Model We use cross-validation to obtain the best cross validation score and c-value.

[49]:
```
# Getting the best cross validation score and our best c-value

C_pool=range(1,31)
best_score=-np.inf
scores = np.zeros(30)


for c in C_pool:
    LR = LogisticRegression(C = c, max_iter = 3000)
    scores[c-1] = cross_val_score(LR, X_train, y_train, cv=10).mean()
    if scores[c-1] > best_score:
        best_score = scores[c-1]
        best_c = c

best_c, best_score
```

[49]:
```
(1, 1.0)
```

[52]:
```
# Creating function to visualize parameter value vs. CV score

def plot_correlation(param,C_pool,scores):
    """
    This function creates a scatterplot of parameter value
    versus CV score

    Parameters
    ----------
    param: str, x-axis of plot

    """
```

```
    fig,ax=plt.subplots(1)
    ax.scatter(C_pool,scores)
    ax.set(xlabel = param, ylabel = "CV score")

# Test function
plot_correlation("c",C_pool,scores)
```



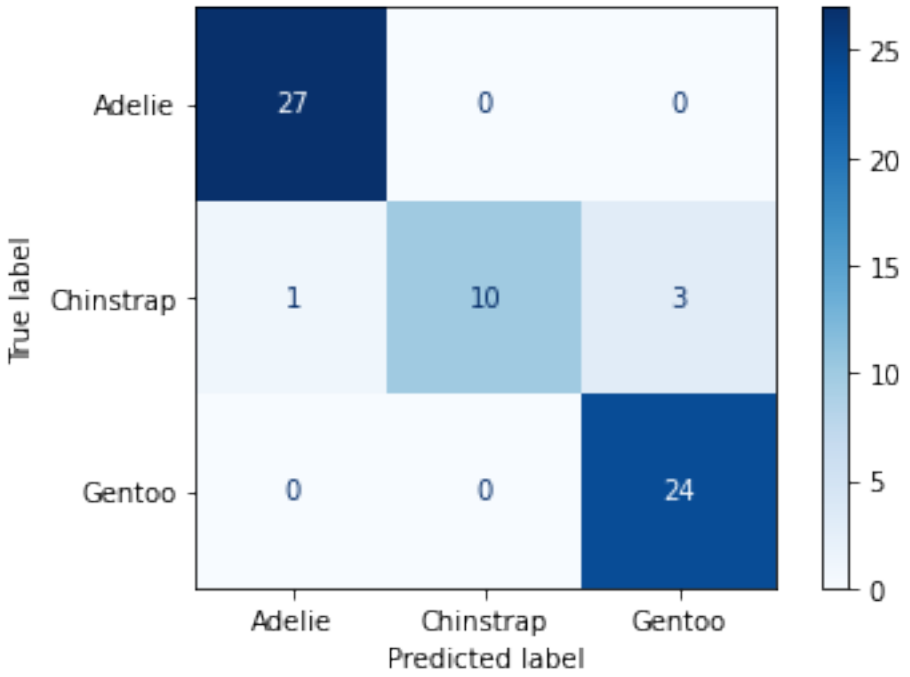
[55]:
```
# Testing CV score with highest cross-validation score on
# unseen data to fit into our model

# Highest CV score
features = ['Island', 'Culmen Length (mm)', 'Culmen Depth (mm)']

# Getting LR model and fitting into model
LR = LogisticRegression(C = 0.1, max_iter = 1000)
LR.fit(X_train[features], y_train)
LR.score(X_train[features], y_train), LR.score(X_test[features], y_test)
```

[55]:
```
(0.9691119691119691, 0.9846153846153847)
```

[151]:
```
# Confusion matrix

from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
```

```
plot_confusion_matrix(LR,
                      X_test[features],
                      y_test,
                      display_labels = ['Adelie', 'Chinstrap', 'Gentoo'],
                      cmap = plt.cm.Blues)
```

[151]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fc08cd876a0>

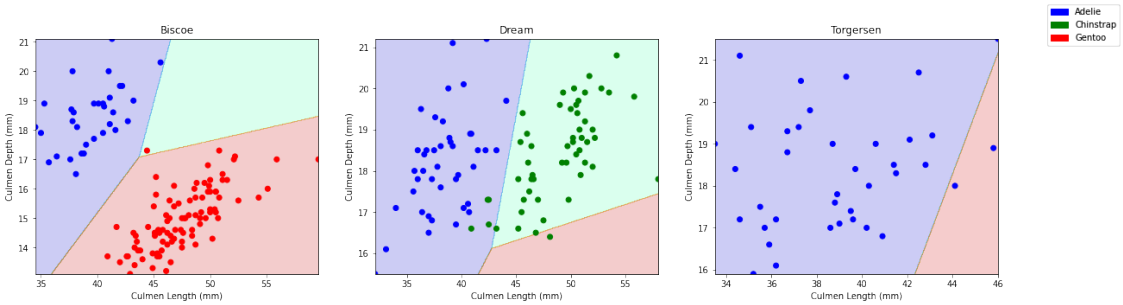


From the confusion matrix, we see that our logistic regression model is mostly accurate; however, there were 3 Adelies predicted as Chinstraps and 1 Chinstrap predicted as an Adelie.

[93]:
```
# Decision Regions

import matplotlib.patches as mpatches

# Assigning variables
quala = features[0]
quanta = features[1]
quantb = features[2]

def plot_regions(c,X,y):
    """
    This function plots the decision regions of a classifiier
```

```
    Parameters
    ----------
    c: classifier
    X: predictor var
    y: target var
    """


    fig,ax=plt.subplots(1,3,figsize=(20,5))
    for island in range(3):
        mask = X[quala] == island
        x0 = X[mask][quanta]
        x1 = X[mask][quantb]
        y0 = y[mask]

        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)
        zz = island * np.ones(501*501)
        XX = xx.ravel()
        YY = yy.ravel()
        ZZ = zz.ravel()

        p = c.predict(np.c_[ZZ,XX,YY])
        p = p.reshape(xx.shape)
        ax[island].contourf(xx, yy, p, cmap = "jet", alpha = .2)

        color_arr = np.array(['blue', 'green', 'red'])
        points = ax[island].scatter(x0,x1,c = [color_arr[i] for i in␣
↪y0],cmap="jet")

        island_name = {
            0 : "Biscoe",
            1 : "Dream",
            2 : "Torgersen"
        }

        ax[island].set(xlabel = quanta, ylabel = quantb)
        ax[island].set(title = island_name[island])
        colors={
            "Adelie" :"blue",
            "Chinstrap" : "green",
            "Gentoo" : "red"
        }
        patches = [mpatches.Patch(color= colors[key], label=key) for key in␣
↪colors]
```

```
        fig.legend(handles = patches)

# Testing function to plot decision regions
plot_regions(LR, X_train, y_train)
```



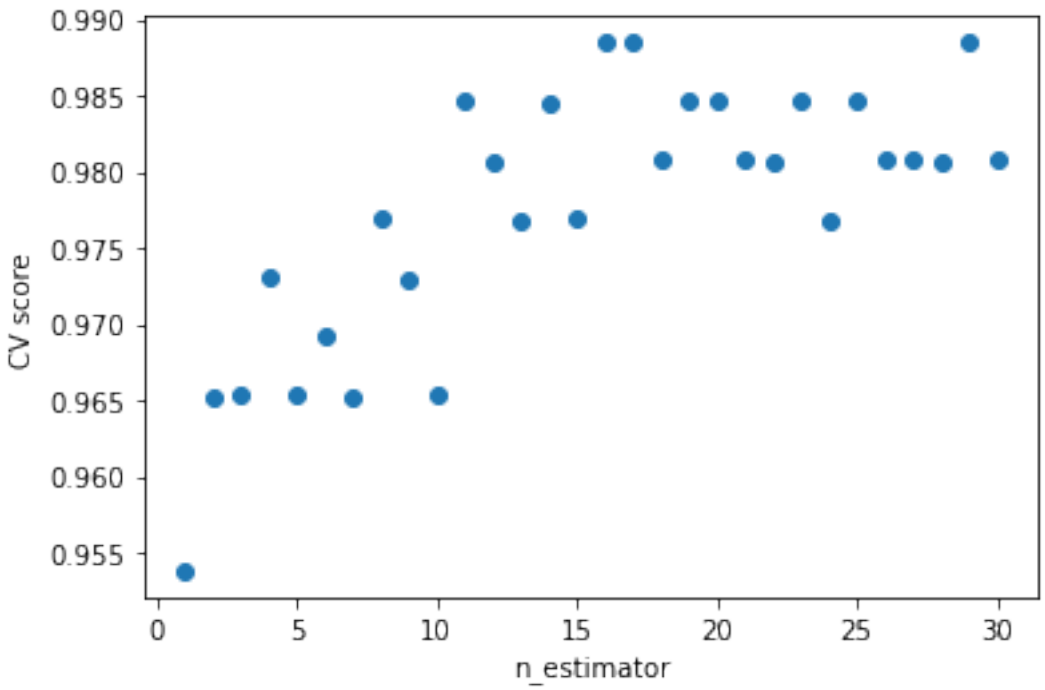
From the graphs, we see that for all three islands most penguins are correctly identified with the exceptions for Biscoe island of one Gentoo being confused for a Chinstrap, Dream island of a few Chinstraps being confused for Adelies and Gentoos, and Torgersen island of two Adelies being confused for Gentoos.

5.2 Random Forest Model

[94]:
```
# Cross-validation to choose complexity parameters

best_score = -np.inf

pool = range(1,31)

scores = np.zeros(30)

for d in pool:
    RF = RandomForestClassifier(n_estimators = d)
    score = cross_val_score(RF, X_train, y_train, cv = 5).mean()
    scores[d-1] = score

    if score > best_score:
        best_n_estimator = d
        best_score = score

best_n_estimator, best_score
```

[94]:
```
(10, 0.9846153846153847)
```

[83]:
```
plot_correlation("n_estimator", pool, scores)
```

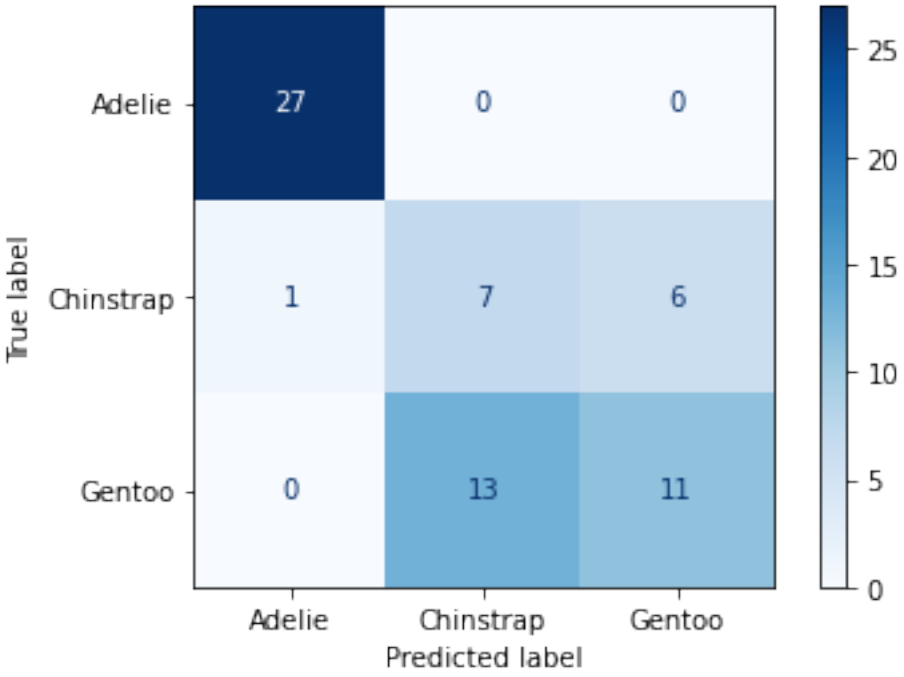


[97]:
```
# Evaluation on unseen testing data
RF = RandomForestClassifier(n_estimators = best_n_estimator)
features = ['Island', 'Culmen Length (mm)', 'Culmen Depth (mm)']
RF.fit(X_train[features], y_train)
RF.score(X_train[features], y_train) , RF.score(X_test[features], y_test)
```

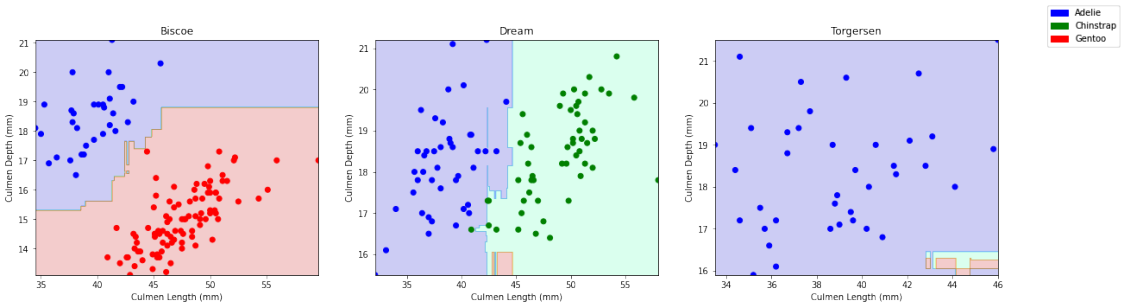
[97]:
```
(1.0, 0.9846153846153847)
```

[152]:
```
# Confusion matrix
plot_confusion_matrix(RF,
                      X_test[features],
                      y_test,
                      display_labels = ['Adelie', 'Chinstrap', 'Gentoo'],
                      cmap = plt.cm.Blues)
```

[152]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7fc08b288be0>
```



[98]:
```
#A visualization of decision regions
plot_regions(RF, X_train, y_train)
```



We see that every penguin is correctly labeled by the decision region with an excepion of a Chinstrap in Dream that was mistakenly identified as Adelie. So far, the random forest model seems to have the best performance.

5.3 K-Neighbor Model Cross validation: We will used different d from 1 to 30 as the parameter "n_neighbors" in KNeighbor model, and record all the cross validation scores we get. The "n_neighbor" with highest score will be "best_k"

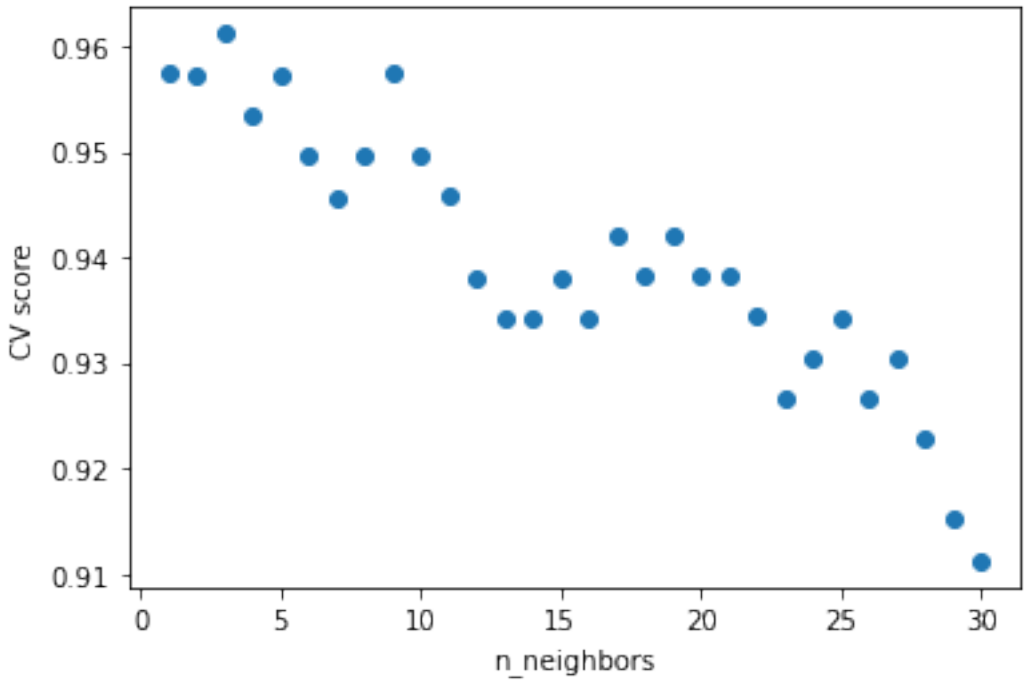
[145]:
```
best_score = -np.inf
best_k = 0
```

```
scores = np.zeros(30)
features = ['Sex', 'Culmen Length (mm)', 'Culmen Depth (mm)']
C_pool = range(1,31)
for d in C_pool:
    KN = KNeighborsClassifier(n_neighbors = d)
    scores[d-1] = cross_val_score(KN, X_train[features], y_train, cv = 5).mean()
    if scores[d-1] > best_score:
        best_score = scores[d-1]
        best_k = d
best_score, best_k
```

[145]: (0.961161387631976, 3)

This plot help visualize the change of cross validation scores for different n_neighbor

[146]:
```
plot_correlation("n_neighbors",C_pool,scores)
```



Next, we use the "best_k" we get in the KN model, fit it with train data, and test it with test data. We print out the scores for both train and test data to ensure there's no severe overfitting issue. The scores we get are ## and ##, which implies the model works effectively.

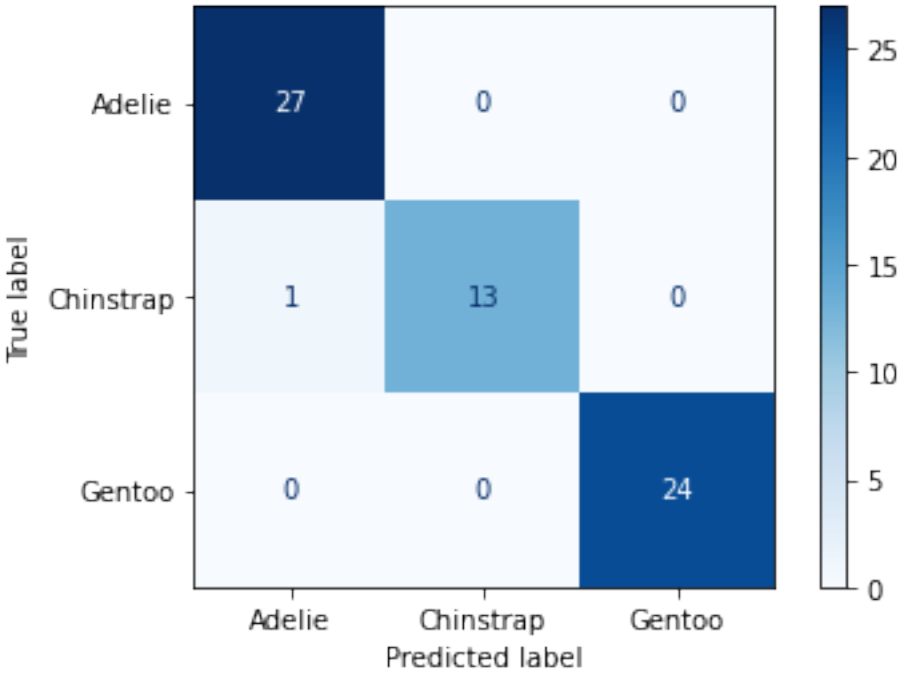
[147]:
```
KN = KNeighborsClassifier(n_neighbors = 3)
KN.fit(X_train[features],y_train)
KN.score(X_train[features],y_train),KN.score(X_test[features],y_test)
```

[147]:
```
(0.9884169884169884, 0.9846153846153847)
```

Then, we will plot the confusion matrix to see how well the model predict the true label of our test data.

[153]:
```
# Confusion matrix
plot_confusion_matrix(KN,
                      X_test[features],
                      y_test,
                      display_labels = ['Adelie', 'Chinstrap', 'Gentoo'],
                      cmap = plt.cm.Blues)
```

[153]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7fc08af98a90>
```

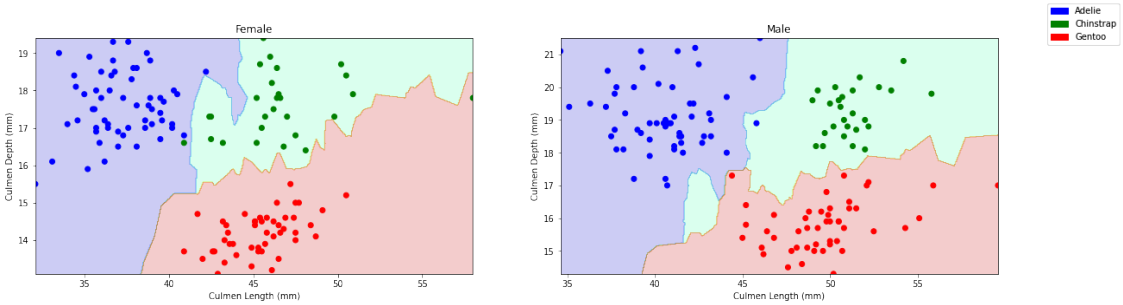


Decision Region: In K Neighbor model, the qualitative predictor is “Sex” instead of “Island” in the other two labels. Thus, we write another function to plot the decision regions. The number of iterations and the labels are different, but all others are the same.

[148]:
```
quali = 'Sex'
quant_1 = 'Culmen Length (mm)'
quant_2 = 'Culmen Depth (mm)'


def plot_KN_regions(c,X,y):
    """
```

```
    Plots the decision regions of KNeighbor classifier
    Parameters:
    c: the classifier
    X: predictor variables
    y: target variable
    """
    sex_name = {0:"Female",1:"Male"}
    fig, ax = plt.subplots(1,2, figsize = (20, 5))

    for sex in range(0,2):
        x0 = X[X[quali] == sex][quant_1]
        x1 = X[X[quali] == sex][quant_2]
        y0 = y[X[quali] == sex]
        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)
        zz = sex * np.ones(501*501)
        XX=xx.ravel()
        YY=yy.ravel()
        ZZ=zz.ravel()

        p=c.predict(np.c_[ZZ,XX,YY])
        p=p.reshape(xx.shape)

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

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

        color = np.array(['blue', 'green', 'red'])

        ax[sex].scatter(x0,x1,c=[color[i] for i in y0],cmap="jet")

        ax[sex].set(xlabel=quant_1,ylabel=quant_2,title = sex_name[sex])

        colors={"Adelie" :"blue", "Chinstrap" : "green", "Gentoo" : "red"}
        patches = [mpatches.Patch(color= colors[key], label=key) for key in␣
↪colors]
        fig.legend(handles = patches)
```

[150]:
```
plot_KN_regions(KN,X_train,y_train)
```



We see that almost every penguins are correctly identified by the decision region with the exceptions of: 2 Female Chinstraps and 1 Male Adelie.

0.7 6. Discussion

The overall performance of our models is quite high. Our Logistic Regression model was able have a model score of approximately 98.5% on the testing sets. However, while almost every penguins are correctly identified by the decision region, our decision regions with the Logistic Regression Model showed with 7 exceptions where the penguins species are not correctly identified. On the other hand, our Random Forest model also had a model score of 98.5% on the testing sets. Moreover, there were only 1 exception where a Chinstrap in Dream that was mistakenly identified as Adelie. And lastly, our K-Neighbor Model also had a mdeol score on 98.5% on the testing sets. And there were 3 penguin that are not correctly identified. So from this testing sets, our Random Forest model had the best performance with the highest accuracy in identifying the penguin species, with a model score of 98.5% on the testing sets.

Our first model used logistic regression and correctly predicted 98.5% of the test data with best c value 1. The model has very simple shapes for its decision regions, indicating less overfitting issue. However, we can see large areas of misprediction comparing to the random forest model. On Dream Island, many of Chinstraps were confused with Adelie. Furthermore, on prediction of the three islands, decision regions corresponding to some species appear even if this speicies doesn't live on the specific island. Our second model is random forest model. we correctly predicted about 98.5% of our penguin species on test set with best n_estimator value 10. The decision regions turn out to be more accurate, with nearly every data points correctly classified. Our third model is the K-Neighbor model, with accuracy of 98.5% and a best_n_neighbors value of 3. In this model, we used a different qualitative variable, which is sex of the penguins. The decision region implies that the model tends to have issue of overfitting, since there's some tiny areas on the graph of Male penguins that is separated from the larger green area. Both female and male include all three species, so in this model we were able to see that K-Neighbor model was able to correctly classify between the three.

The models could be improved if we had used the sex, culmen depth, and culmen length columns instead of island, culmen length and culmen depth for our logistic regression and random forest models, since these columns had a higher CV score. If more or different data were available for our model, i.e. more predictor variables, it would allow us to have better accuracy in our models, although we did obtain pretty accuarate results with the models we created.