The Data

We will be using the same dataset through our discussions on classification with tree-methods (Decision Tree,Random Forests, and Boosted Trees) in order to compare performance metrics across these related models.

"Palmer Penguins" dataset Summary:

- penguins_size.csv: Simplified data from original penguin data sets. Contains variables:
  - species: penguin species (Chinstrap, Adélie, or Gentoo)
  - culmen_length_mm: culmen length (mm)
  - culmen_depth_mm: culmen depth (mm)
  - flipper_length_mm: flipper length (mm)
  - body_mass_g: body mass (g)
  - island: island name (Dream, Torgersen, or Biscoe) in the Palmer Archipelago (Antarctica)
  - sex: penguin sex

Note: The culmen is "the upper ridge of a bird's beak"

Our goal is to create a model that can help predict a species of a penguin based on physical attributes, then we can use that model to help researchers classify penguins in the field, instead of needing an experienced biologist

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv("penguins_size.csv")
df.head()
```

Out[2]:

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FE
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FE

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   species            344 non-null    object 
 1   island             344 non-null    object 
 2   culmen_length_mm   342 non-null    float64
 3   culmen_depth_mm    342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                334 non-null    object 
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

Missing Data

Recall the purpose is to create a model for future use, so data points missing crucial information won't help in this task, especially since for future data points we will assume the research will grab the relevant feature information.

In [4]:
```
df.isna().sum()
```

Out[4]:
```
species               0
island                0
culmen_length_mm      2
culmen_depth_mm       2
flipper_length_mm     2
body_mass_g           2
sex                  10
dtype: int64
```

In [5]:
```
# What percentage are we dropping?
100*(10/344)
```

Out[5]:
```
2.9069767441860463
```

In [6]:
```
df = df.dropna()
df.shape
```

Out[6]:
```
(334, 7)
```

In [7]:
```
df['sex'].unique()
```

Out[7]:
```
array(['MALE', 'FEMALE', '.'], dtype=object)
```

In [8]:
```
df = df[df['sex']!='.']
df.shape
```

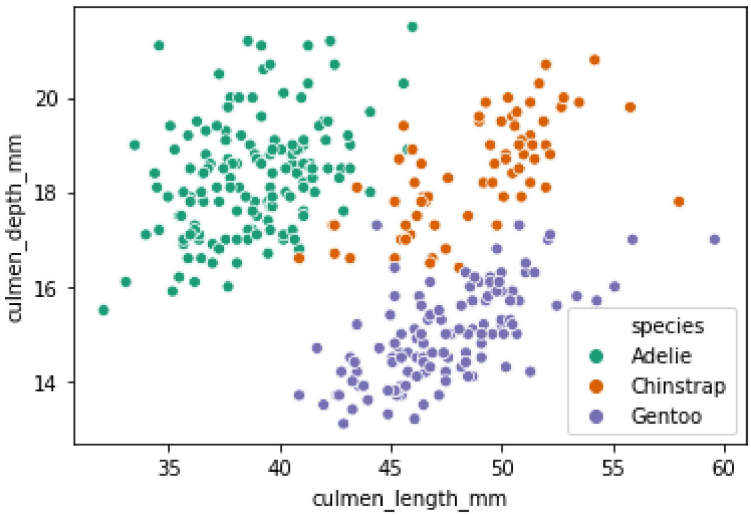
Out[8]:
```
(333, 7)
```

In [9]:
```
df['island'].unique()
```

Out[9]:
```
array(['Torgersen', 'Biscoe', 'Dream'], dtype=object)
```

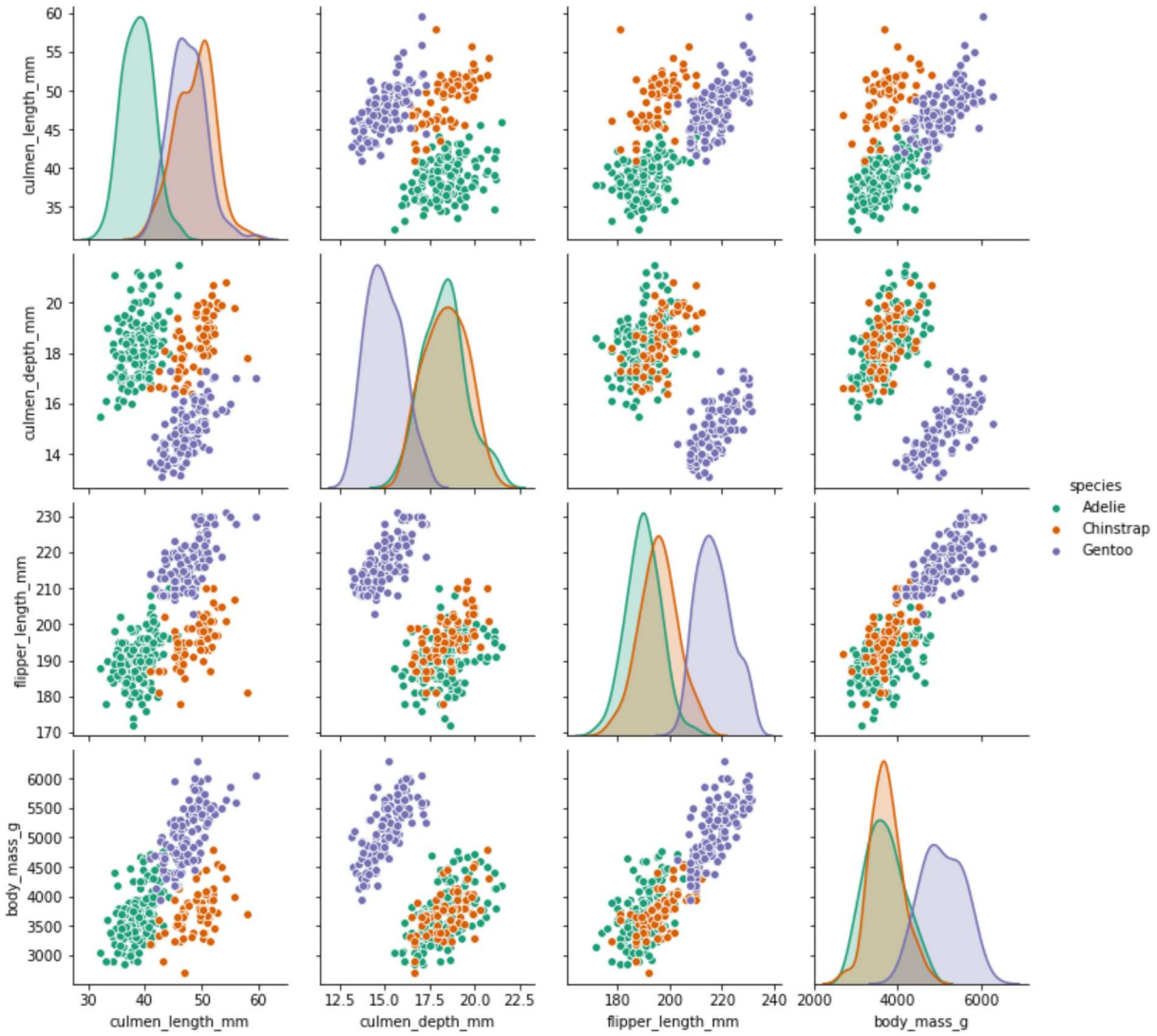
In [10]:
```
sns.scatterplot(x='culmen_length_mm',y='culmen_depth_mm',data=df,hue='species',pa
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2cf0f999970>
```



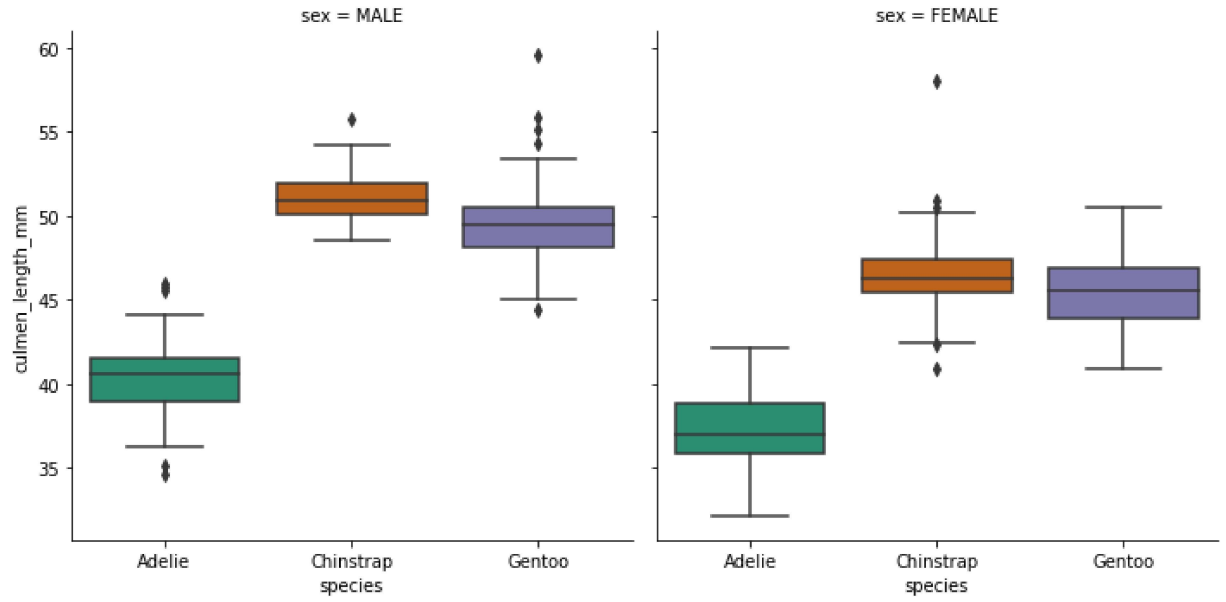
In [11]:
```
sns.pairplot(df,hue='species',palette='Dark2')
```

Out[11]:
```
<seaborn.axisgrid.PairGrid at 0x2cf10132670>
```



In [12]:
```
sns.catplot(x='species',y='culmen_length_mm',data=df,kind='box',col='sex',palette
```

Out[12]:
```
<seaborn.axisgrid.FacetGrid at 0x2cf109406a0>
```



Feature Engineering

In [13]:
```
pd.get_dummies(df.drop('species',axis=1),drop_first=True)
```

Out[13]:

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	island_Dream	islan
0	39.1	18.7	181.0	3750.0	0	
1	39.5	17.4	186.0	3800.0	0	
2	40.3	18.0	195.0	3250.0	0	
4	36.7	19.3	193.0	3450.0	0	
5	39.3	20.6	190.0	3650.0	0	
...	...	...	...	...	...	
338	47.2	13.7	214.0	4925.0	0	
340	46.8	14.3	215.0	4850.0	0	
341	50.4	15.7	222.0	5750.0	0	
342	45.2	14.8	212.0	5200.0	0	
343	49.9	16.1	213.0	5400.0	0	

333 rows × 7 columns

In [14]:
```
X = pd.get_dummies(df.drop('species',axis=1),drop_first=True)
y = df['species']
```

Train | Test Split

In [15]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_s
```

Decision Tree Classifier - with default Hyperparameters

In [16]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [17]:
```
model = DecisionTreeClassifier()
```

In [18]:
```
model.fit(X_train,y_train)
```

Out[18]:
```
DecisionTreeClassifier()
```

In [19]:
```
base_pred = model.predict(X_test)
```

Evaluation

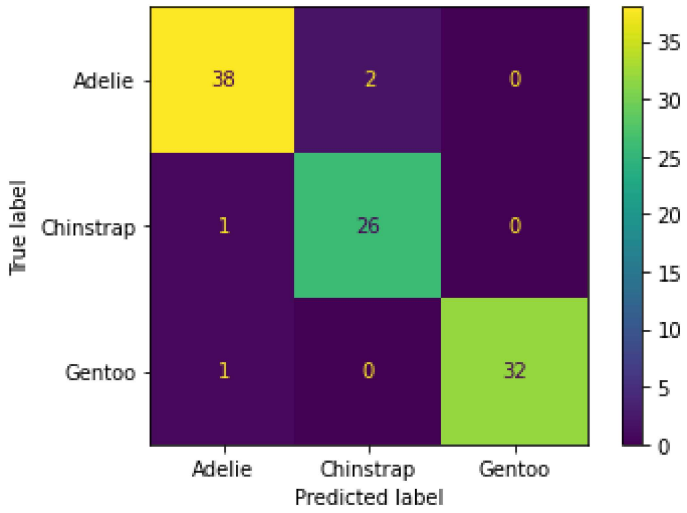
In [20]:
```
from sklearn.metrics import confusion_matrix,classification_report,plot_confusion
```

In [21]:
```
confusion_matrix(y_test,base_pred)
```

Out[21]:
```
array([[38,  2,  0],
       [ 1, 26,  0],
       [ 1,  0, 32]], dtype=int64)
```

In [22]:
```
plot_confusion_matrix(model,X_test,y_test)
```

Out[22]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2cf10f0f310
>
```



In [23]:
```
print(classification_report(y_test,base_pred))
```

```
              precision    recall  f1-score   support

      Adelie       0.95      0.95      0.95        40
   Chinstrap       0.93      0.96      0.95        27
      Gentoo       1.00      0.97      0.98        33

    accuracy                           0.96       100
   macro avg       0.96      0.96      0.96       100
weighted avg       0.96      0.96      0.96       100
```

In [24]:
```
model.feature_importances_
```

Out[24]:
```
array([0.33350103, 0.04582245, 0.57575804, 0.        , 0.03806069,
       0.00685778, 0.        ])
```

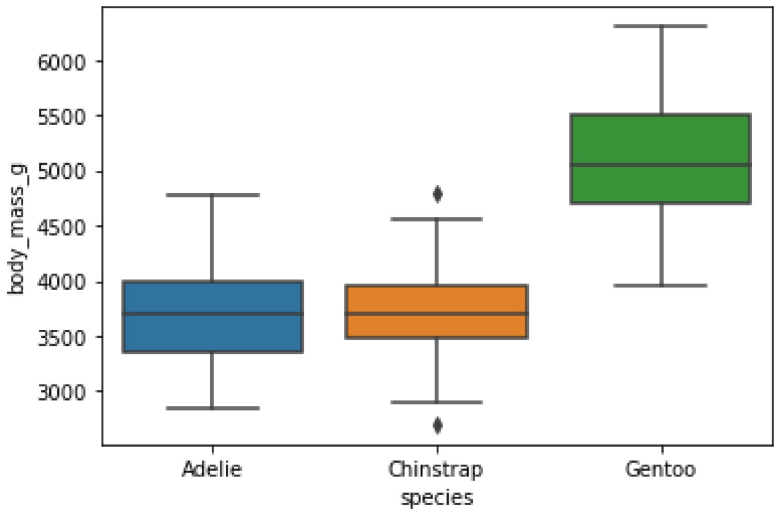
In [25]:
```
pd.DataFrame(index=X.columns,data=model.feature_importances_,columns=['Feature Im
```

Out[25]:

	Feature Importance
culmen_length_mm	0.333501
culmen_depth_mm	0.045822
flipper_length_mm	0.575758
body_mass_g	0.000000
island_Dream	0.038061
island_Torgersen	0.006858
sex_MALE	0.000000

In [26]:
```
sns.boxplot(x='species',y='body_mass_g',data=df)
```

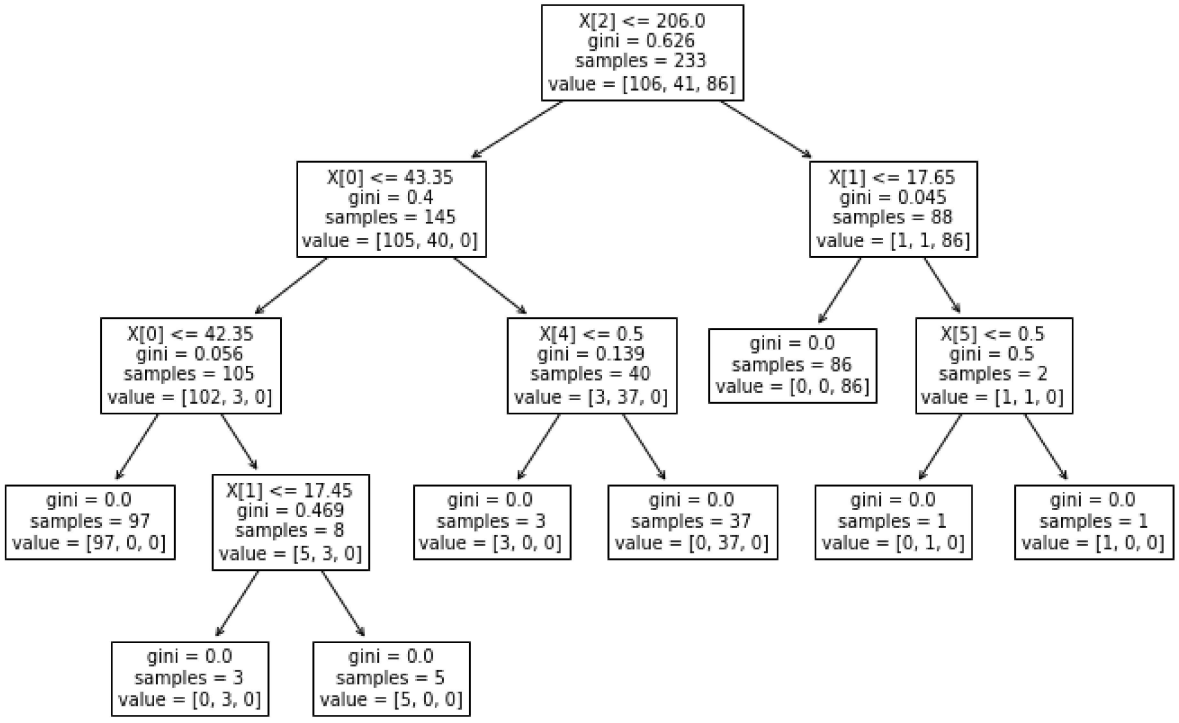
Out[26]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2cf108f9220>
```



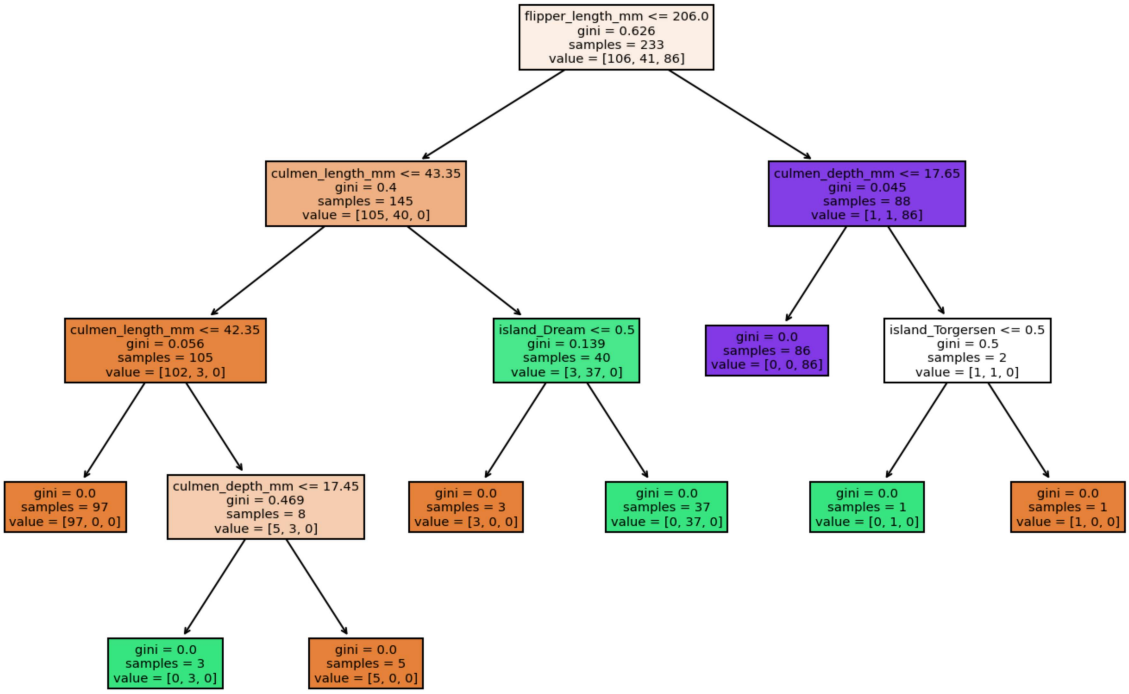
Visualize the Tree

In [27]:
```
from sklearn.tree import plot_tree
```

In [29]:
```
plt.figure(figsize=(12,8))
plot_tree(model)
plt.show()
```



In [30]:
```
plt.figure(figsize=(12,8),dpi=150)
plot_tree(model,filled=True,feature_names=X.columns)
plt.show()
```



Reporting Model Results

To begin experimenting with hyperparameters, let's create a function that reports back classification results and plots out the tree.

In [31]:
```
def report_model(model):
    model_preds = model.predict(X_test)
    print(classification_report(y_test,model_preds))
    print('\n')
    plt.figure(figsize=(12,8),dpi=150)
    plot_tree(model,filled=True,feature_names=X.columns)
```

Understanding Hyperparameters

Max Depth

In [32]:
```
pruned_tree = DecisionTreeClassifier(max_depth=2)
pruned_tree.fit(X_train,y_train)
```

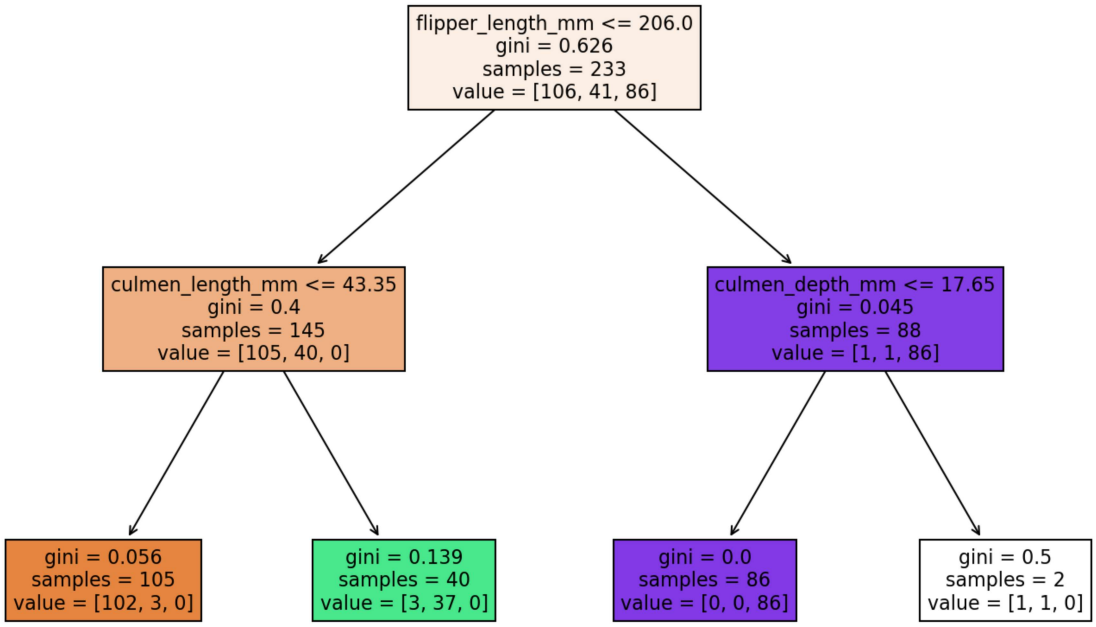
Out[32]:
```
DecisionTreeClassifier(max_depth=2)
```

In [33]:
```
report_model(pruned_tree)
```

```
              precision    recall  f1-score   support

      Adelie       0.87      0.97      0.92        40
   Chinstrap       0.91      0.78      0.84        27
      Gentoo       1.00      0.97      0.98        33

    accuracy                           0.92       100
   macro avg       0.93      0.91      0.91       100
weighted avg       0.92      0.92      0.92       100
```



Max Leaf Nodes

In [34]:
```
pruned_tree = DecisionTreeClassifier(max_leaf_nodes=3)
pruned_tree.fit(X_train,y_train)
```

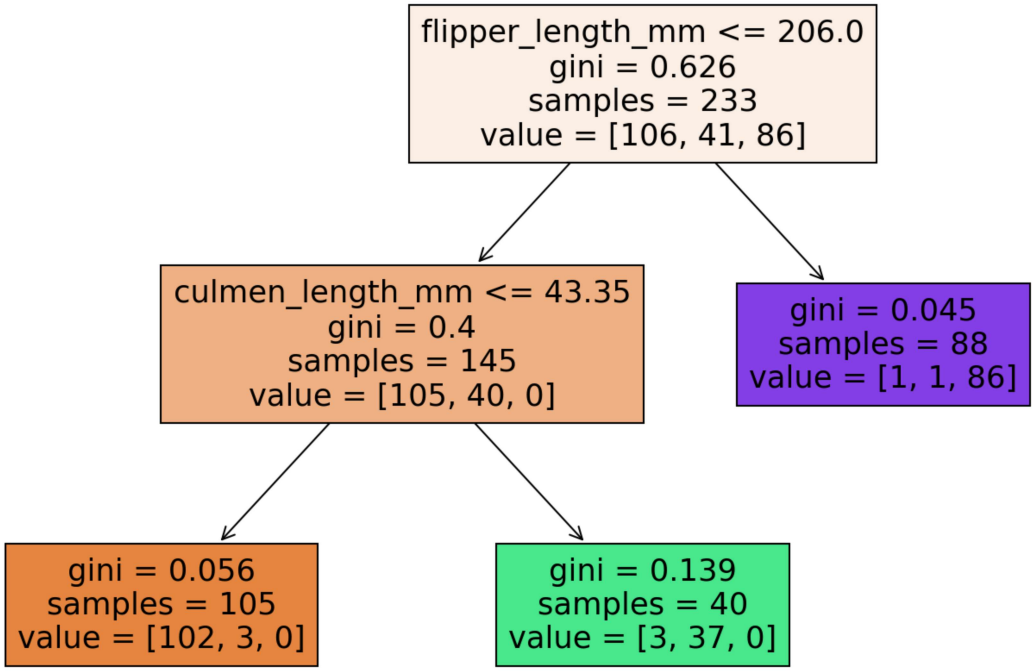
Out[34]:
```
DecisionTreeClassifier(max_leaf_nodes=3)
```

In [35]:
```
report_model(pruned_tree)
```

```
              precision    recall  f1-score   support

      Adelie       0.95      0.95      0.95        40
   Chinstrap       0.91      0.78      0.84        27
      Gentoo       0.86      0.97      0.91        33

    accuracy                           0.91       100
   macro avg       0.91      0.90      0.90       100
weighted avg       0.91      0.91      0.91       100
```



Criterion

In [36]:
```
entropy_tree = DecisionTreeClassifier(criterion='entropy')
entropy_tree.fit(X_train,y_train)
```

Out[36]:
```
DecisionTreeClassifier(criterion='entropy')
```

In [37]:
```
report_model(entropy_tree)
```

```
              precision    recall  f1-score   support

      Adelie       0.86      0.95      0.90        40
   Chinstrap       0.92      0.81      0.86        27
      Gentoo       1.00      0.97      0.98        33

    accuracy                           0.92       100
   macro avg       0.93      0.91      0.92       100
weighted avg       0.92      0.92      0.92       100
```