Decision Trees

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

The Data

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

We will work with the "Palmer Penguins" dataset, as it is simple enough to help us fully understand how changing hyperparameters can change classification results.

Data were collected and made available by Dr. Kristen Gorman and the Palmer Station, Antarctica LTER, a member of the Long Term Ecological Research Network.

Gorman KB, Williams TD, Fraser WR (2014) Ecological Sexual Dimorphism and Environmental Variability within a Community of Antarctic Penguins (Genus Pygoscelis). PLoS ONE 9(3): e90081. doi:10.1371/journal.pone.0090081

Summary: The data folder contains two CSV files. For intro courses/examples, you probably want to use the first one (penguins_size.csv).

- **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**
- **(Not used) penguins_lter.csv: Original combined data for 3 penguin species**

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 [132]:

```
df = pd.read_csv("D:\\Study\\Programming\\python\\Python course from udemy\\Udemy - 2022 Python for Machine Learning & Data Science Masterclass\\01 - Introduction to Course\\1UNZIP-FOR-NOTEBOOKS-FINAL\\DATA\\penguins_size.csv")
df.head()
```

Out[132]:

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

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex

EDA

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 [133]:

```
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
```

In [134]:

```
df.isnull().sum()
```

Out[134]:

```
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 [135]:

```
# What percentage are we dropping?
100*(10/344)
```

Out[135]:

```
2.9069767441860463
```

In [136]:

```
df= df.dropna()
```

In [137]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 334 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   species            334 non-null    object 
 1   island             334 non-null    object 
 2   culmen_length_mm   334 non-null    float64
 3   culmen_depth_mm    334 non-null    float64
```

```
 4   flipper_length_mm  334 non-null    float64
 5   body_mass_g        334 non-null    float64
 6   sex                334 non-null    object
dtypes: float64(4), object(3)
memory usage: 20.9+ KB
```

In [138]:

```
df.head()
```

Out[138]:

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	MALE

In [139]:

```
df['island'].unique()
```

Out[139]:

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

In [140]:

```
df['sex'].unique()
```

Out[140]:

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

In [141]:

```
df[df['sex']=='.']
```

Out[141]:

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
336	Gentoo	Biscoe	44.5	15.7	217.0	4875.0	.

In [142]:

```
# Here if we want to drop that row
#df=df[df['sex']!='.']
#df
```

In [143]:

```
# Here we will do some analyis and try to find out the sex of missing penguin
# That pengine belongs to 'Gentoo' species
df[df['species']=='Gentoo'].groupby('sex').describe().transpose()
```

Out[143]:

	sex	.	FEMALE	MALE
culmen_length_mm	**count**	1.0	58.000000	61.000000
	mean	44.5	45.563793	49.473770
	std	NaN	2.051247	2.720594
	min	44.5	40.900000	44.400000
	25%	44.5	43.850000	48.100000

	sex	.	FEMALE	MALE
	50%	44.5	45.500000	49.500000
	75%	44.5	46.875000	50.500000
	max	44.5	50.500000	59.600000
culmen_depth_mm	count	1.0	58.000000	61.000000
	mean	15.7	14.237931	15.718033
	std	NaN	0.540249	0.741060
	min	15.7	13.100000	14.100000
	25%	15.7	13.800000	15.200000
	50%	15.7	14.250000	15.700000
	75%	15.7	14.600000	16.100000
	max	15.7	15.500000	17.300000
flipper_length_mm	count	1.0	58.000000	61.000000
	mean	217.0	212.706897	221.540984
	std	NaN	3.897856	5.673252
	min	217.0	203.000000	208.000000
	25%	217.0	210.000000	218.000000
	50%	217.0	212.000000	221.000000
	75%	217.0	215.000000	225.000000
	max	217.0	222.000000	231.000000
body_mass_g	count	1.0	58.000000	61.000000
	mean	4875.0	4679.741379	5484.836066
	std	NaN	281.578294	313.158596
	min	4875.0	3950.000000	4750.000000
	25%	4875.0	4462.500000	5300.000000
	50%	4875.0	4700.000000	5500.000000
	75%	4875.0	4875.000000	5700.000000
	max	4875.0	5200.000000	6300.000000

By observing above data we find that 336 is female

In [144]:

```
# here we assign that 336 is female
df.at[336,'sex']= 'FEMALE'
```

In [145]:

```
df.loc[336]
```

Out[145]:

```
species              Gentoo
island               Biscoe
culmen_length_mm       44.5
culmen_depth_mm        15.7
flipper_length_mm     217.0
body_mass_g          4875.0
sex                  FEMALE
Name: 336, dtype: object
```

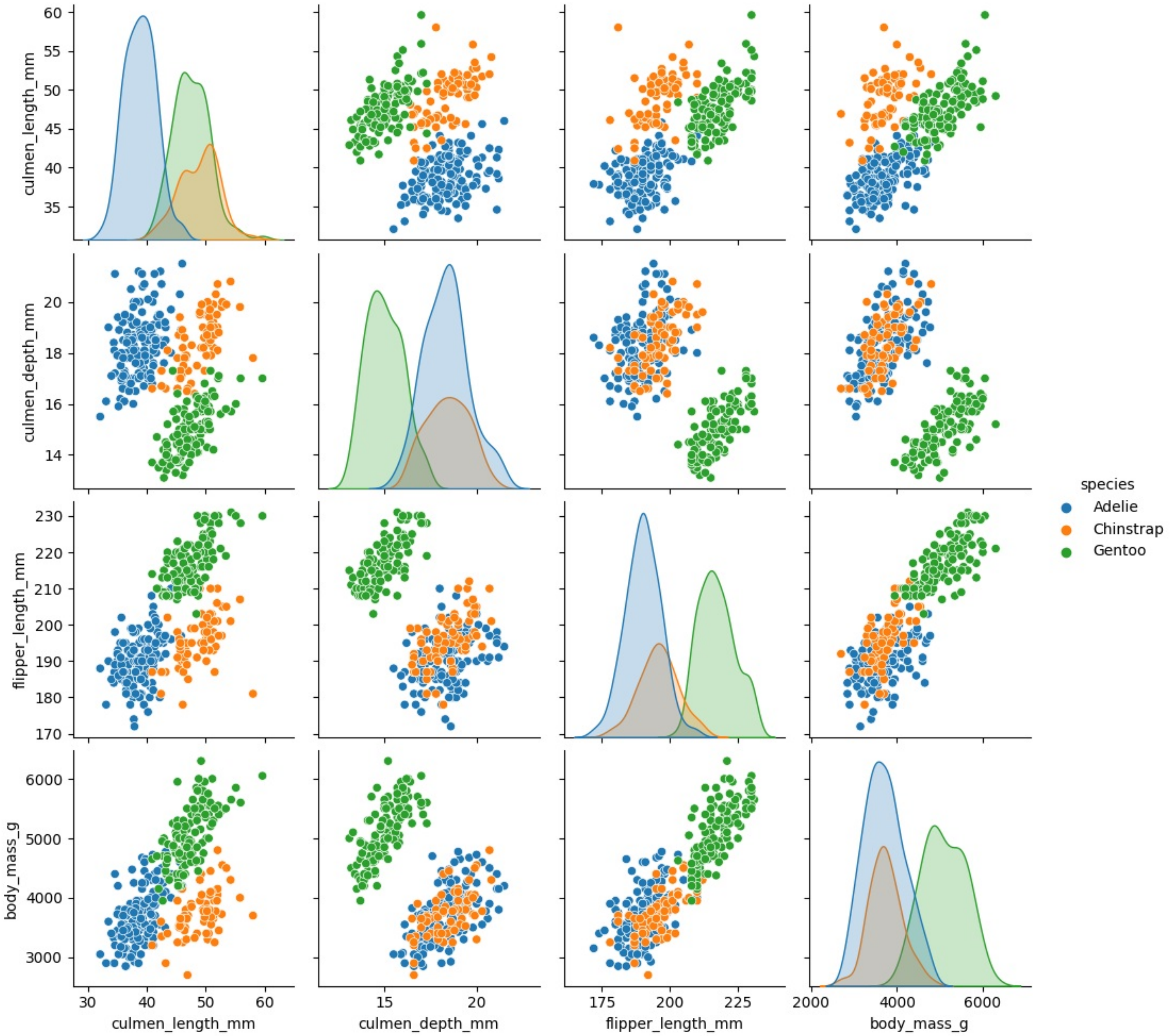
Visualization

In [146]:

```
sns.pairplot(data=df,hue='species')
```

Out[146]:

```
<seaborn.axisgrid.PairGrid at 0x1630e218520>
```

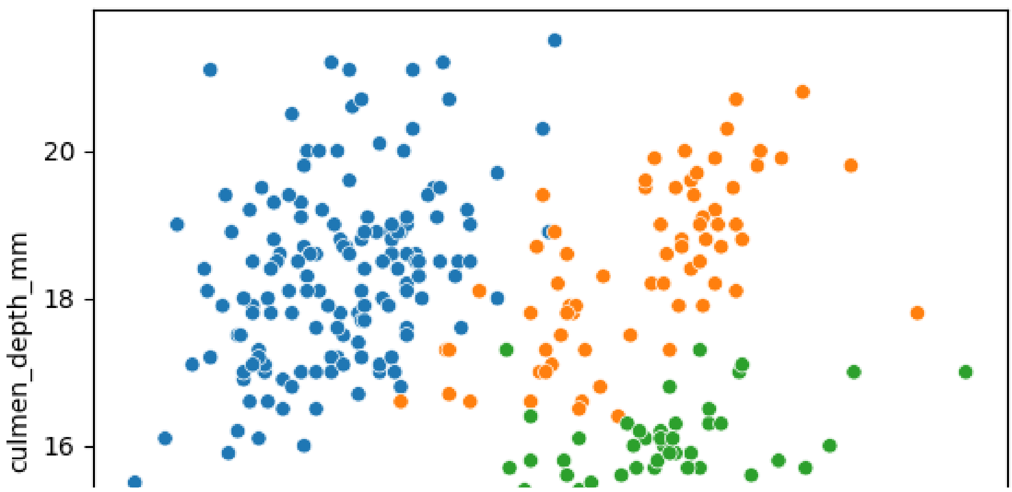


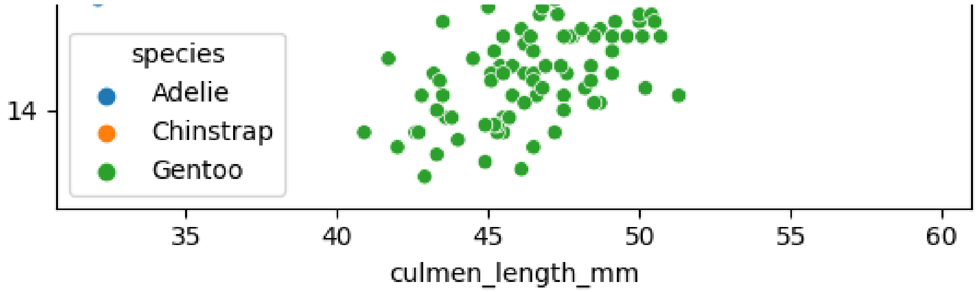
In [147]:

```
sns.scatterplot(x='culmen_length_mm',y='culmen_depth_mm',data=df,hue='species')
```

Out[147]:

```
<AxesSubplot: xlabel='culmen_length_mm', ylabel='culmen_depth_mm'>
```



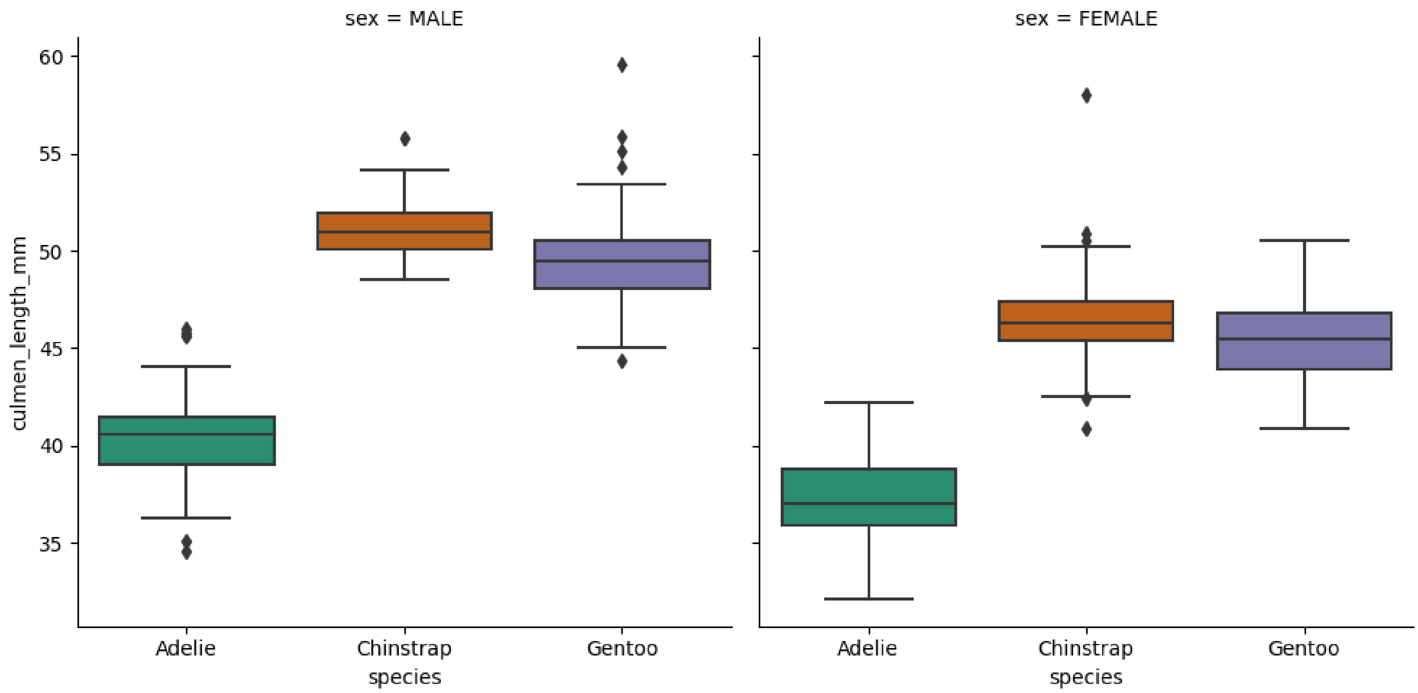


In [148]:

```
sns.catplot(x='species',y='culmen_length_mm',data=df,kind='box',col='sex',palette='Dark2
')
```

Out[148]:

```
<seaborn.axisgrid.FacetGrid at 0x16330ff9f10>
```



Feature Engineering

Here we have different different island so we need to create dummy for that

In [149]:

```
pd.get_dummies(df).head()
```

Out[149]:

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	species_Adelie	species_Chinstrap	species_Gento
0	39.1	18.7	181.0	3750.0	1	0	
1	39.5	17.4	186.0	3800.0	1	0	
2	40.3	18.0	195.0	3250.0	1	0	
4	36.7	19.3	193.0	3450.0	1	0	
5	39.3	20.6	190.0	3650.0	1	0	

In [150]:

```
pd.get_dummies(df.drop('species',axis=1),drop_first=True).head()
```

Out[150]:

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	island_Dream	island_Torgersen	sex_MALE
0	39.1	18.7	181.0	3750.0	0	1	1
1	39.5	17.4	186.0	3800.0	0	1	0
2	40.3	18.0	195.0	3250.0	0	1	0
4	36.7	19.3	193.0	3450.0	0	1	0
5	39.3	20.6	190.0	3650.0	0	1	1

Train | Test Split

In [151]:

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

In [152]:

```
y.head()
```

Out[152]:

```
0    Adelie
1    Adelie
2    Adelie
4    Adelie
5    Adelie
Name: species, dtype: object
```

In [153]:

```
from sklearn.model_selection import train_test_split
```

In [154]:

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=101)
```

Decision Tree Classifier

Default Hyperparameters

In [155]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [156]:

```
model = DecisionTreeClassifier()
```

In [157]:

```
model.fit(X_train,y_train)
```

Out[157]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [158]:

```
base_pred = model.predict(X_test)
```

Evaluation

In [159]:

```
from sklearn.metrics import confusion_matrix,classification_report
```

In [160]:

```
from mlxtend.plotting import plot_confusion_matrix
```

In [164]:

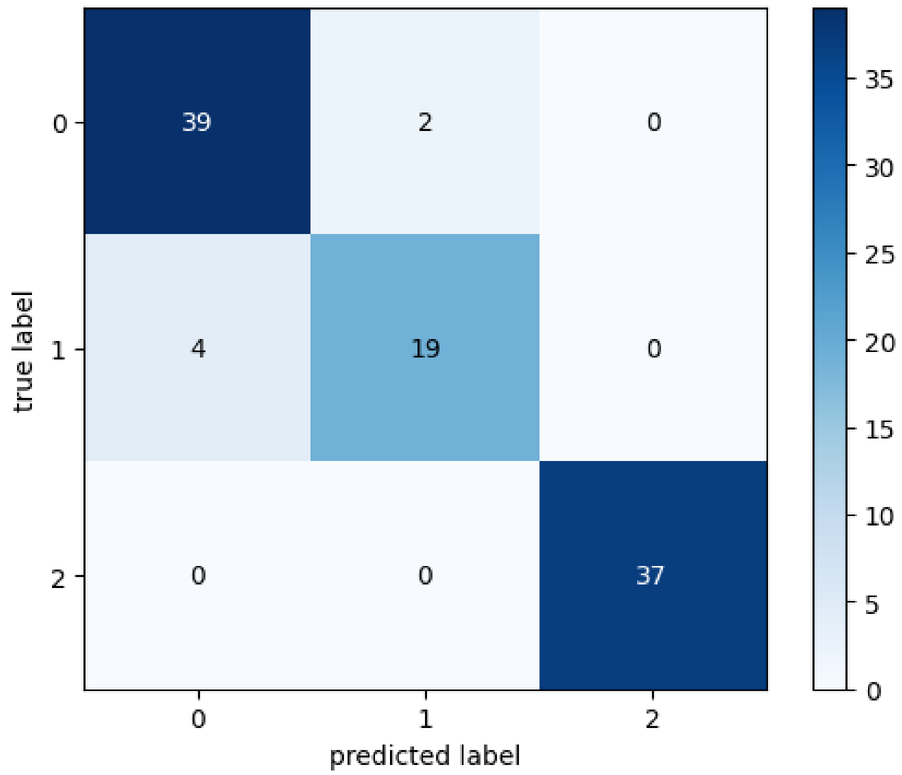
```
gd=confusion_matrix(y_test,base_pred)
gd
```

Out[164]:

```
array([[39,  2,  0],
       [ 4, 19,  0],
       [ 0,  0, 37]], dtype=int64)
```

In [167]:

```
fig, ax = plot_confusion_matrix(conf_mat=gd,colorbar=True)
plt.show()
```



In [168]:

```
print(classification_report(y_test,base_pred))
```

```
              precision    recall  f1-score   support

      Adelie       0.91      0.95      0.93        41
   Chinstrap       0.90      0.83      0.86        23
      Gentoo       1.00      1.00      1.00        37

    accuracy                           0.94       101
   macro avg       0.94      0.93      0.93       101
weighted avg       0.94      0.94      0.94       101
```

In [172]:

```
# These are columns as per there importance in decision making
```

```
model.feature_importances_
```

Out[172]:

```
array([0.33514864, 0.05221421, 0.53120101, 0.01325074, 0.0681854 ,
       0.        , 0.        ])
```

In [183]:

```
X.columns
```

Out[183]:

```
Index(['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm',
       'body_mass_g', 'island_Dream', 'island_Torgersen', 'sex_MALE'],
      dtype='object')
```

In [174]:

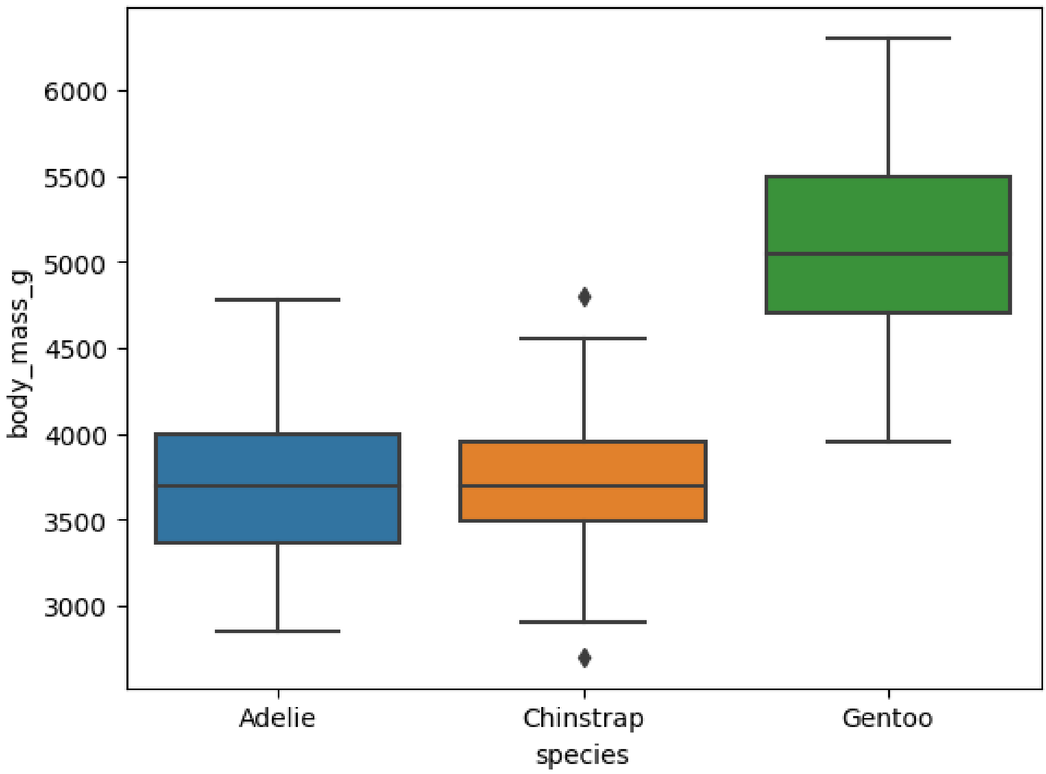
```
# Lets create a DateFrame so that we can see those things properly
pd.DataFrame(index=X.columns, data=model.feature_importances_,columns=['Feature Importan
ce'])
```

Out[174]:

	Feature Importance
culmen_length_mm	0.335149
culmen_depth_mm	0.052214
flipper_length_mm	0.531201
body_mass_g	0.013251
island_Dream	0.068185
island_Torgersen	0.000000
sex_MALE	0.000000

In [176]:

```
sns.boxplot(data=df,x='species',y='body_mass_g');
```



Visualize the Tree

This function is fairly new, you may want to review the online docs:

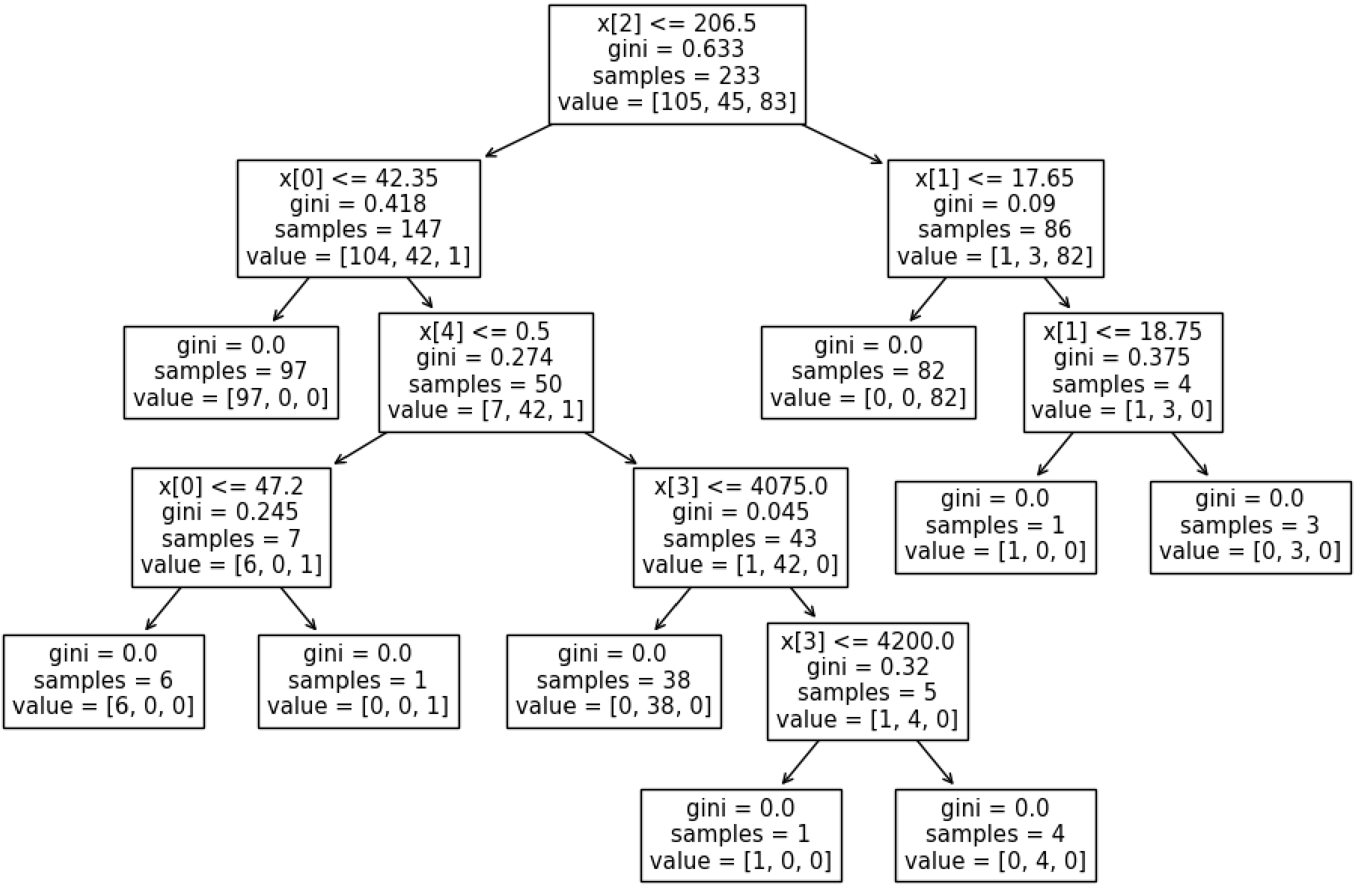
Online Documentation: https://scikit-learn.org/stable/modules/generated/sklearn.tree.plot_tree.html

In [179]:

```
from sklearn.tree import plot_tree
```

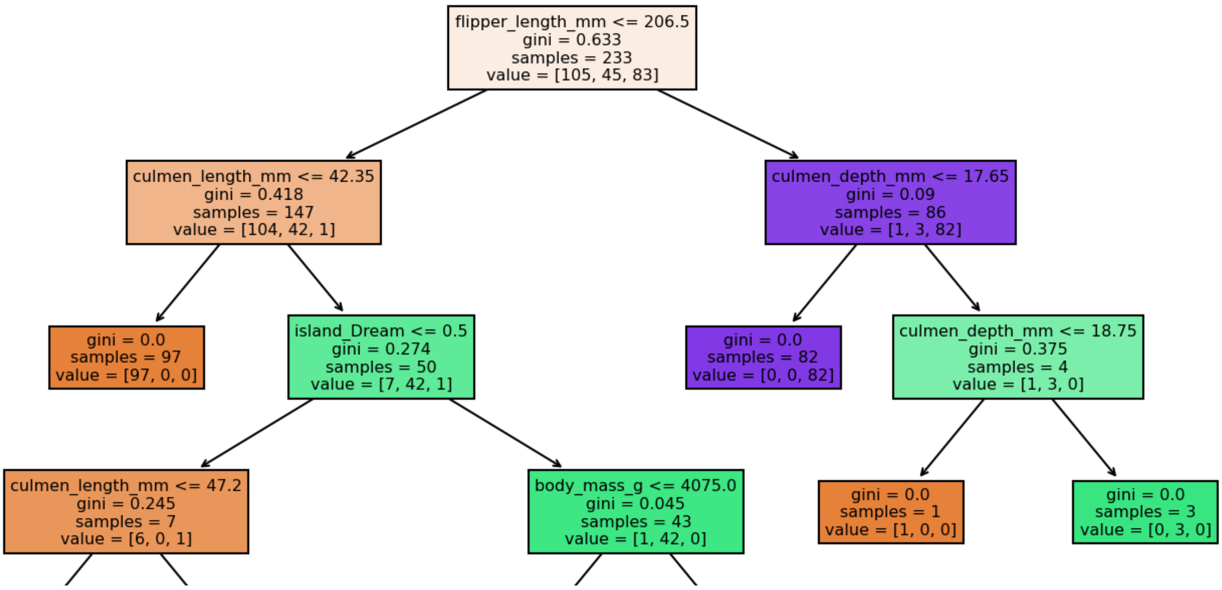
In [181]:

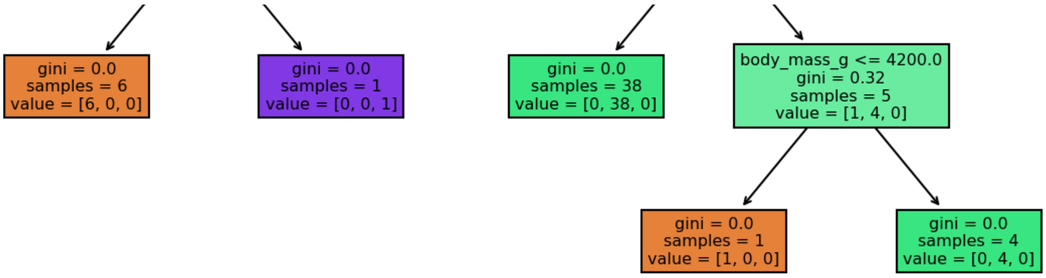
```
plt.figure(figsize=(12,8),dpi=100)
plot_tree(model);
```



In [186]:

```
# Adding filled colors and feature_names in boxes
plt.figure(figsize=(12,8),dpi=150)
plot_tree(model,filled=True,feature_names=X.columns);
```





Reporting Model Results

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

In [190]:

```
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);
```

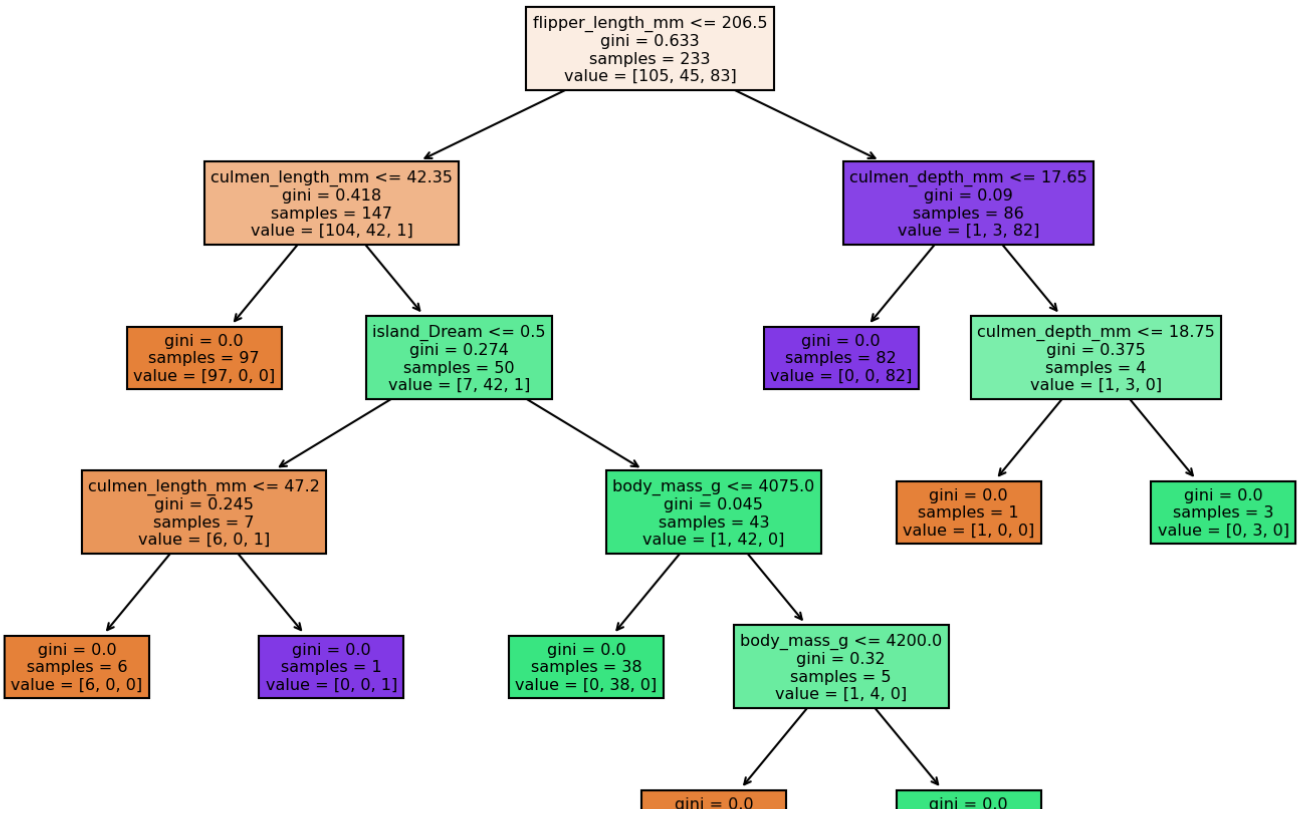
In [192]:

```
report_model(model)
```

```
              precision    recall  f1-score   support

      Adelie       0.91      0.95      0.93        41
   Chinstrap       0.90      0.83      0.86        23
      Gentoo       1.00      1.00      1.00        37

    accuracy                           0.94       101
   macro avg       0.94      0.93      0.93       101
weighted avg       0.94      0.94      0.94       101
```



Understanding Hyperparameters

Max Depth

In [191]:

```
#help(DecisionTreeClassifier)
```

In [193]:

```
pruned_tree = DecisionTreeClassifier(max_depth=2)
pruned_tree.fit(X_train,y_train)
```

Out[193]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=2)
```

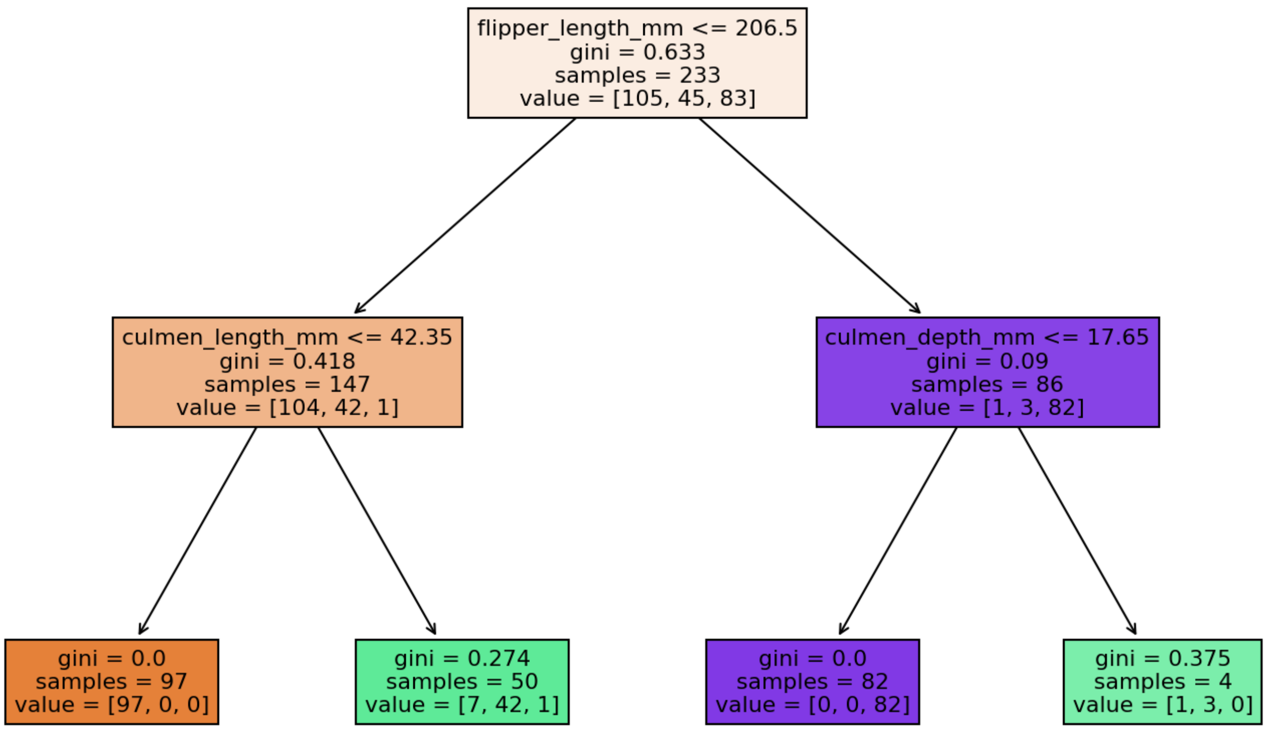
In [194]:

```
report_model(pruned_tree)
```

```
              precision    recall  f1-score   support

      Adelie       0.97      0.88      0.92        41
   Chinstrap       0.81      0.96      0.88        23
      Gentoo       1.00      1.00      1.00        37

    accuracy                           0.94       101
   macro avg       0.93      0.94      0.93       101
weighted avg       0.95      0.94      0.94       101
```



In [196]:

```
pruned_tree = DecisionTreeClassifier(max_leaf_nodes=3)
pruned_tree.fit(X_train,y_train)
```

Out[196]:

```
▾          DecisionTreeClassifier
DecisionTreeClassifier(max_leaf_nodes=3)
```

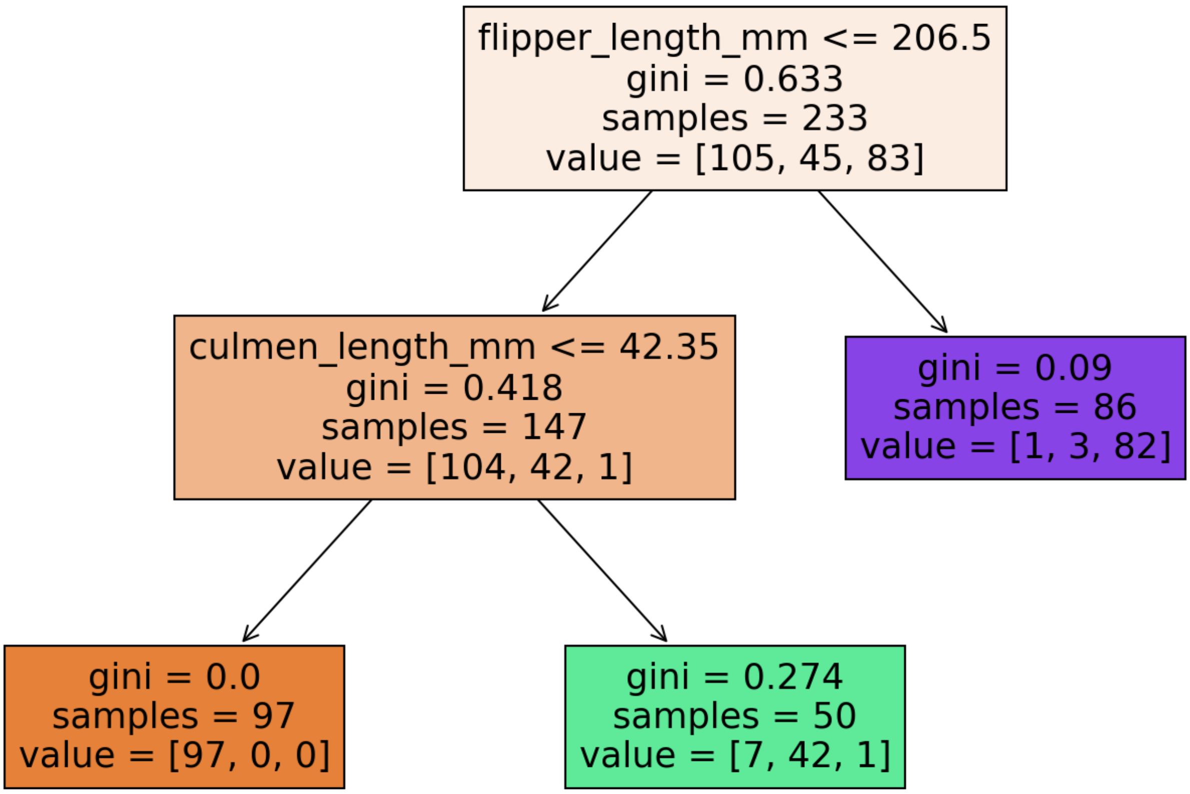
In [197]:

```
report_model(pruned_tree)
```

```
              precision    recall  f1-score   support

      Adelie       0.97      0.88      0.92        41
   Chinstrap       0.83      0.87      0.85        23
      Gentoo       0.93      1.00      0.96        37

    accuracy                           0.92       101
   macro avg       0.91      0.92      0.91       101
weighted avg       0.92      0.92      0.92       101
```



Criterion

In [199]:

```
entropy_tree = DecisionTreeClassifier(criterion='entropy')
entropy_tree.fit(X_train,y_train)
```

Out[199]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy')
```

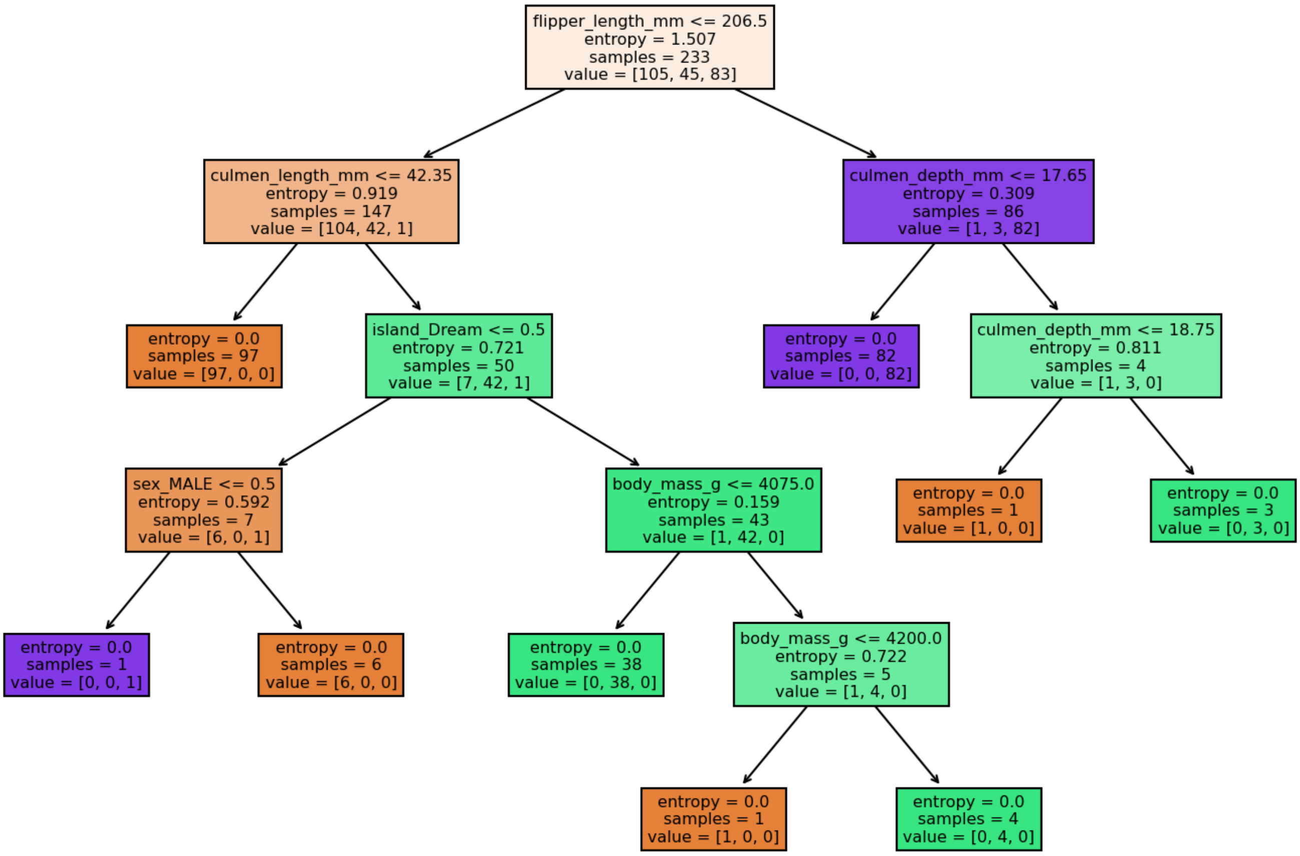
In [200]:

```
report_model(entropy_tree)
```

```
              precision    recall  f1-score   support

      Adelie       0.91      0.95      0.93        41
   Chinstrap       0.90      0.83      0.86        23
      Gentoo       1.00      1.00      1.00        37

    accuracy                           0.94       101
   macro avg       0.94      0.93      0.93       101
weighted avg       0.94      0.94      0.94       101
```



In [ ]: