Business Problem

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
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0
3	Adelie	Torgersen	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0

penguins_size.csv: Simplified data from original penguin data sets.

- 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

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
```

EDA

In [4]:
```
df['species'].unique()
```

Out[4]:
```
array(['Adelie', 'Chinstrap', 'Gentoo'], dtype=object)
```

In [5]:
```
df['species'].value_counts()
```

Out[5]:
```
Adelie       152
Gentoo       124
Chinstrap     68
Name: species, dtype: int64
```

In [6]:
```
df['island'].unique()
```

Out[6]:
```
array(['Torgersen', 'Biscoe', 'Dream'], dtype=object)
```

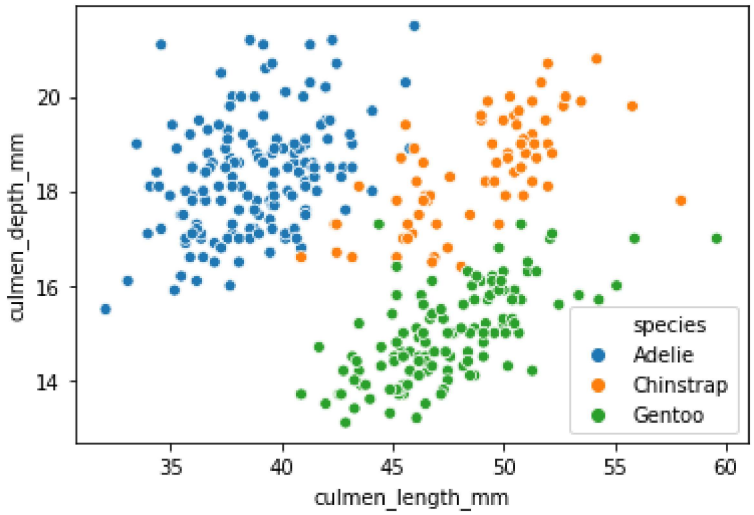
In [7]:
```
df['sex'].unique()
```

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

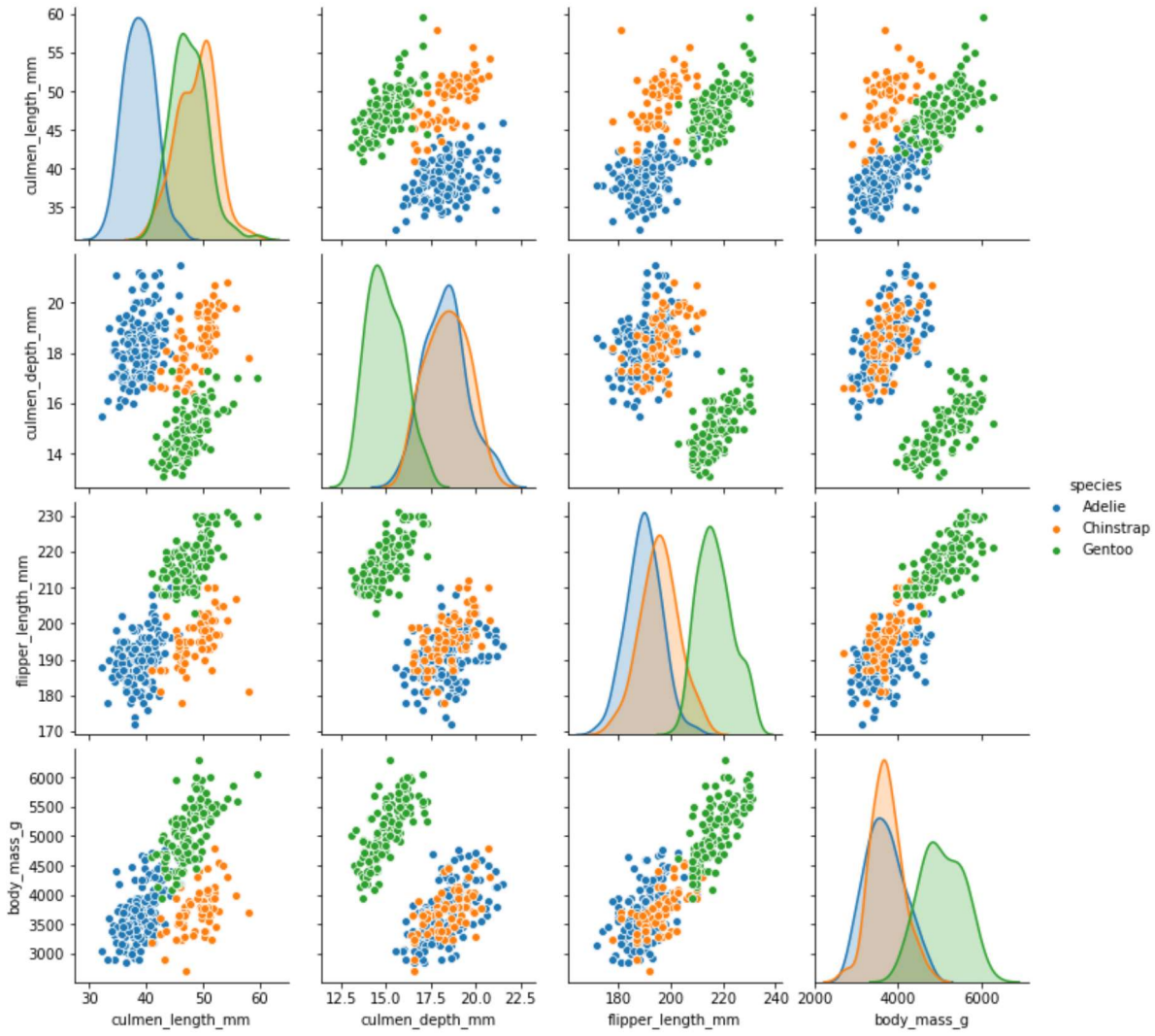
In [8]:
```
df = df[df['sex']!='.']
df.shape
```

Out[8]:
```
(343, 7)
```

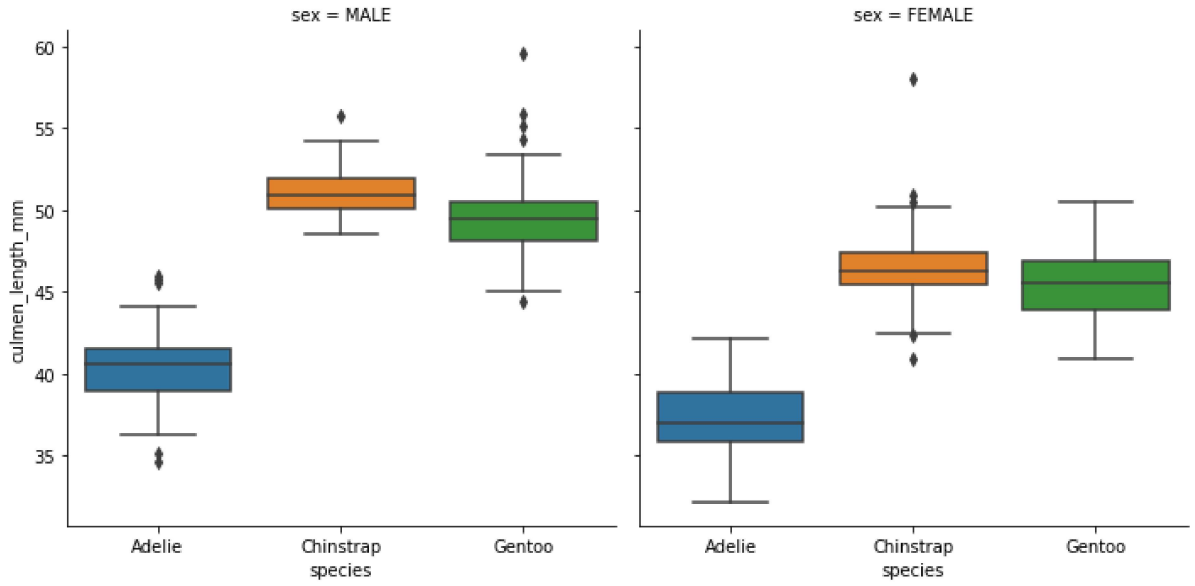
In [9]:
```
sns.scatterplot(x='culmen_length_mm',y='culmen_depth_mm',data=df,hue='species'
)
plt.show()
```



In [10]:
```
sns.pairplot(df,hue='species')
plt.show()
```



In [11]:
```
sns.catplot(x='species',y='culmen_length_mm',data=df,kind='box',col='sex')
plt.show()
```



Feature Engineering

Missing Values

In [12]:
```
df.isna().sum()
```

Out[12]:
```
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 [13]:
```
df = df.dropna()
df.shape
```

Out[13]:
```
(333, 7)
```

Encoding

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

Out[14]:

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	island_Dream	isl
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 [15]:
```
X = pd.get_dummies(df.drop('species',axis=1),drop_first=True)
y = df['species']
```

Train/Test Split

In [16]:
```
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, rando
m_state=101)
```

Modelling

Decision Tree Classifier - with default Hyperparameters

In [17]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [18]:
```
model = DecisionTreeClassifier()
```

In [19]:
```
model.fit(X_train,y_train)
```

Out[19]:
```
DecisionTreeClassifier()
```

Prediction

In [20]:
```
base_pred = model.predict(X_test)
```

In [21]:
```
pred_train = model.predict(X_train)
```

Evaluation

In [22]:
```
from sklearn.metrics import classification_report,plot_confusion_matrix,accuracy_score
```

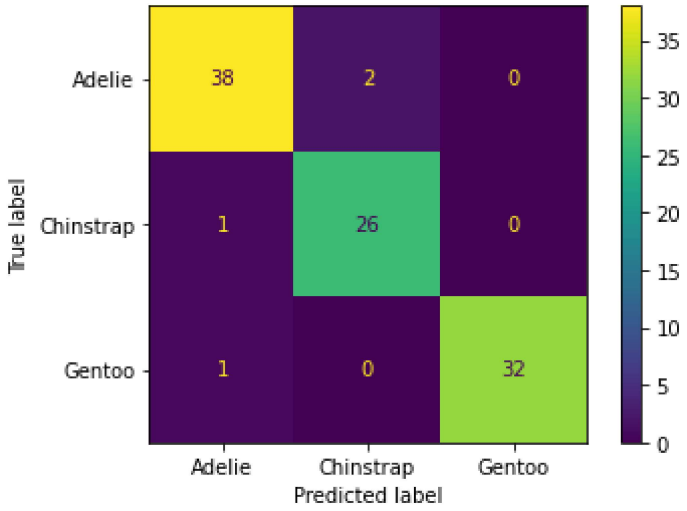
In [23]:
```
accuracy_score(y_train,pred_train)
```

Out[23]: 1.0

In [24]:
```
accuracy_score(y_test,base_pred)
```

Out[24]: 0.96

In [25]:
```
plot_confusion_matrix(model,X_test,y_test)
plt.show()
```



In [26]:
```
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 [27]:
```
model.feature_importances_
```

Out[27]:
```
array([0.33350103, 0.02010577, 0.57575804, 0.        , 0.03806069,
       0.00685778, 0.02571668])
```

In [28]:
```
pd.DataFrame(index=X.columns,data=model.feature_importances_,columns=['Feature Importance'])
```

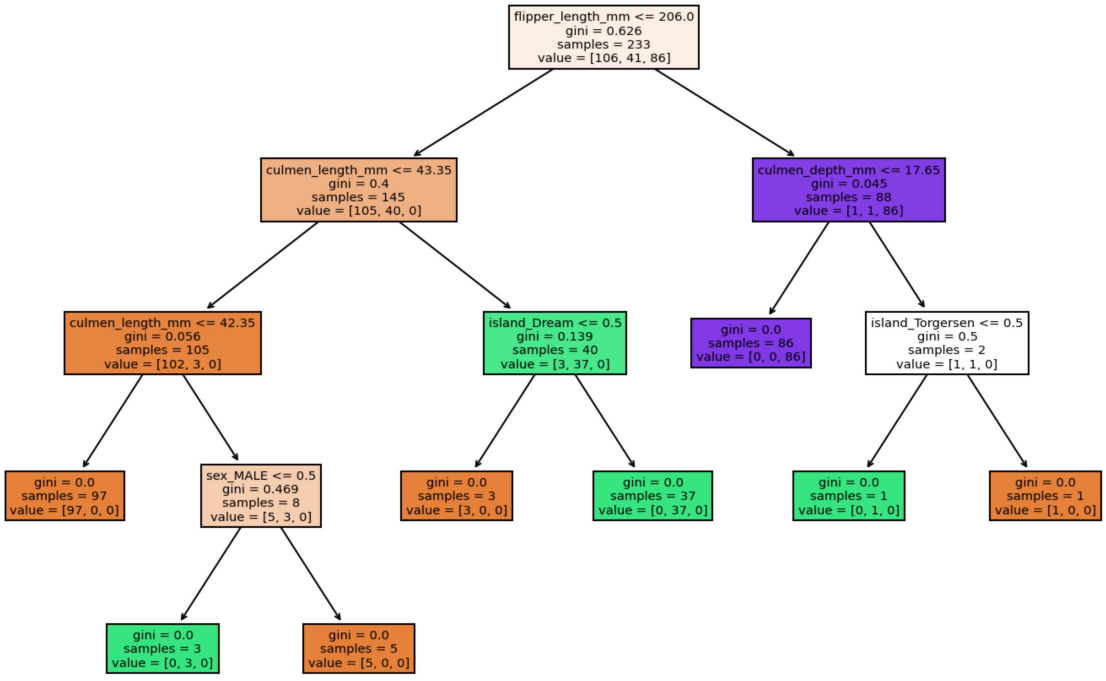
Out[28]:

	Feature Importance
culmen_length_mm	0.333501
culmen_depth_mm	0.020106
flipper_length_mm	0.575758
body_mass_g	0.000000
island_Dream	0.038061
island_Torgersen	0.006858
sex_MALE	0.025717

Visualize the Tree

In [29]:
```
from sklearn.tree import plot_tree
```

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



Optimization -- Pruning of Decision Tree

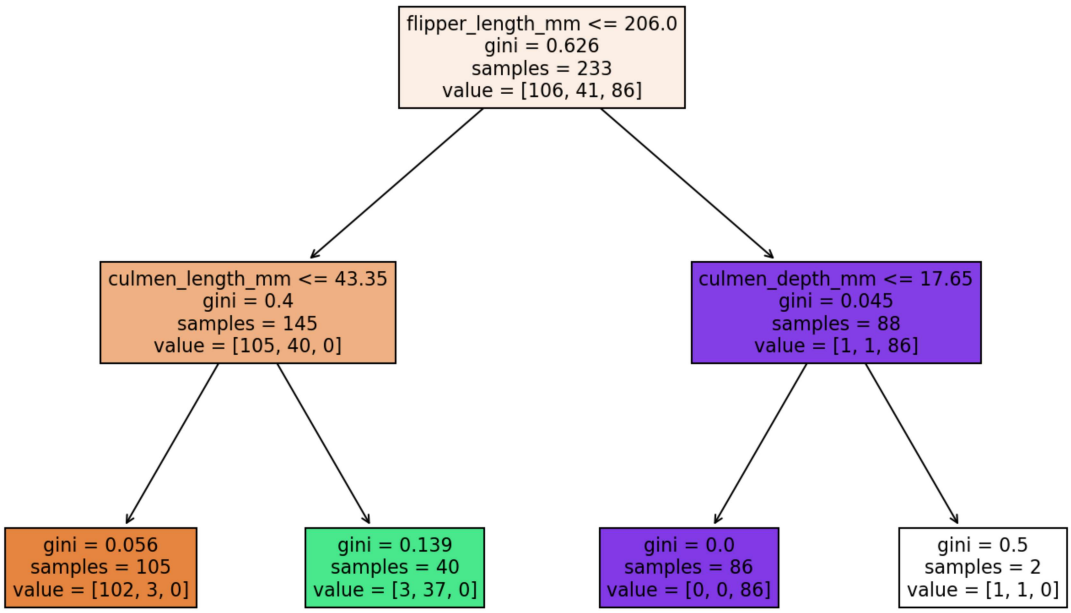
Max Depth

In [31]:
```
pruned_tree = DecisionTreeClassifier(max_depth=2)
```

In [32]:
```
def report_model(model):
    model.fit(X_train,y_train)
    model_preds = model.predict(X_test)
    pred_train = model.predict(X_train)
    print("Train Accuracy:",accuracy_score(y_train,pred_train))
    print("Test Accuracy:",accuracy_score(y_test,model_preds))
    plt.figure(figsize=(12,8),dpi=150)
    plot_tree(model,filled=True,feature_names=X.columns)
```

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

```
Train Accuracy: 0.9699570815450643
Test Accuracy: 0.92
```



Max Leaf Nodes

In [34]:
```
pruned_tree = DecisionTreeClassifier(max_leaf_nodes=3)
```

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

```
Train Accuracy: 0.9656652360515021
Test Accuracy: 0.91
```