The Data

"Palmer Penguins" dataset Summary:

- 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

Describe

In [3]:
```
df.shape
```

Out[3]:
```
(344, 7)
```

In [4]:
```
df.columns
```

Out[4]:
```
Index(['species', 'island', 'culmen_length_mm', 'culmen_depth_mm',
       'flipper_length_mm', 'body_mass_g', 'sex'],
      dtype='object')
```

In [5]:
```
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 [6]:
```
df.isnull().sum()
```

Out[6]:
```
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 [7]:
```
df.describe()
```

Out[7]:

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
count	342.000000	342.000000	342.000000	342.000000
mean	43.921930	17.151170	200.915205	4201.754386
std	5.459584	1.974793	14.061714	801.954536
min	32.100000	13.100000	172.000000	2700.000000
25%	39.225000	15.600000	190.000000	3550.000000
50%	44.450000	17.300000	197.000000	4050.000000
75%	48.500000	18.700000	213.000000	4750.000000
max	59.600000	21.500000	231.000000	6300.000000

EDA

Feature Engineering

In [8]:
```
#Droping the missing values
df = df.dropna()
```

In [9]:
```
#shape of data after dropping missing values
df.shape
```

Out[9]:
```
(334, 7)
```

X & y

In [10]:
```
X = pd.get_dummies(df.drop('species',axis=1),drop_first=True)
X
```

Out[10]:

	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	

334 rows × 8 columns

In [11]:
```
y = df['species']
```

Train Test Split

In [12]:
```
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
```

Modelling

- Import Classifier
- Save with "model name" with default parameters
- Fit the model
- Predict on X_test & y_pred_test
- Predict on X_ train & y_ pred_ train

In [13]:
```
# Import Classifier
from sklearn.ensemble import RandomForestClassifier
```

In [14]:
```
help(RandomForestClassifier)
```

...

In [15]:
```
# Save with "model name" with default parameters
model = RandomForestClassifier(n_estimators=10,random_state=101) # Use 10 random
```

In [16]:
```
# Fit the model
model.fit(X_train,y_train)
```

Out[16]:
```
RandomForestClassifier(n_estimators=10, random_state=101)
```

In [17]:
```
# Predict on X_test & y_pred_test
y_pred_test = model.predict(X_test)
```

In [18]:
```
# Predict on X_train & y_pred_train
y_pred_train = model.predict(X_train)
```

Evaluation

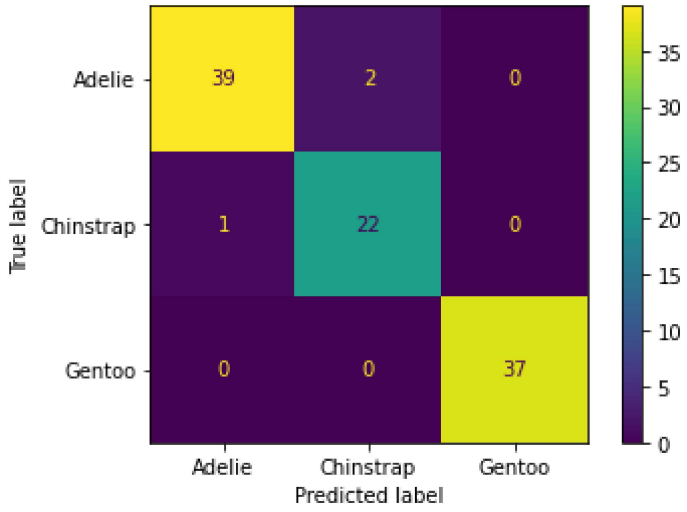
In [19]:
```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_pred_train,y_train))
print(accuracy_score(y_pred_test,y_test))
```

```
1.0
0.9702970297029703
```

In [20]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred_test)
```

Out[20]:
```
array([[39,  2,  0],
       [ 1, 22,  0],
       [ 0,  0, 37]], dtype=int64)
```

In [21]:
```
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(model,X_test,y_test)
plt.show()
```



In [22]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred_test))
```

```
              precision    recall  f1-score   support

      Adelie       0.97      0.95      0.96        41
   Chinstrap       0.92      0.96      0.94        23
      Gentoo       1.00      1.00      1.00        37

    accuracy                           0.97       101
   macro avg       0.96      0.97      0.97       101
weighted avg       0.97      0.97      0.97       101
```

In [23]:
```
# from sklearn.metrics import confusion_matrix,classification_report,plot_confusi
```

Feature Importance

Very useful attribute of the trained model!

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

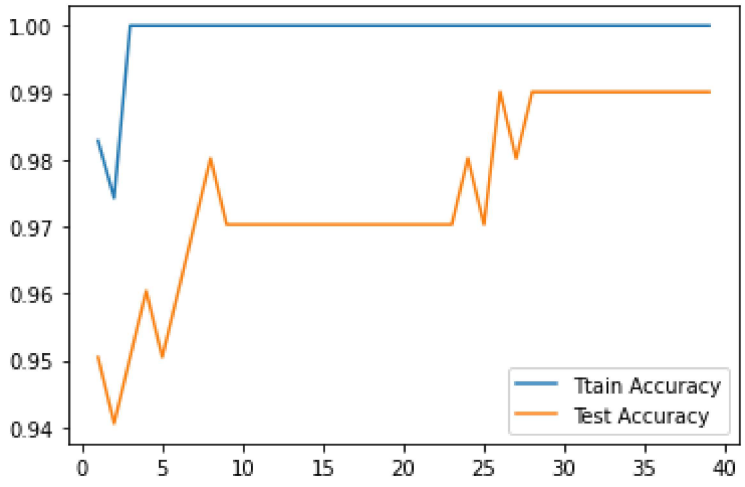
Out[24]:
```
array([0.31867744, 0.1018487 , 0.17343398, 0.21316964, 0.14512091,
       0.03720114, 0.00632264, 0.00422556])
```

Choosing correct number of trees

- Let's explore if continually adding more trees improves performance...

In [25]:
```
# Let's plot out accuracy vs. Number of Estimators

test_acc = []
train_acc=[]


for n in range(1,40):
    # Use n random trees
    model = RandomForestClassifier(n_estimators=n,max_features='auto',bootstrap=T
    model.fit(X_train,y_train)

    train_preds = model.predict(X_train)
    train_acc.append(accuracy_score(train_preds,y_train))

    test_preds = model.predict(X_test)
    test_acc.append(accuracy_score(test_preds,y_test))
```

In [26]:
```
# Let's plot out accuracy vs. Number of Estimators

plt.plot(range(1,40),train_acc,label='Ttain Accuracy')
plt.plot(range(1,40),test_acc,label='Test Accuracy')

plt.legend()
plt.show()
```



Random Forest - HyperParameter Exploration

In [27]:
```
from sklearn.model_selection import GridSearchCV
```

In [28]:
```
param_grid = {'n_estimators':[26,64,100,128,200], 'max_features':[2,3,4]}
```

In [29]:
```
rfc = RandomForestClassifier()

grid = GridSearchCV(rfc,param_grid,cv=5,scoring="accuracy")

grid.fit(X_train,y_train)
```

Out[29]:
```
GridSearchCV(cv=5, estimator=RandomForestClassifier(),
             param_grid={'max_features': [2, 3, 4],
                         'n_estimators': [26, 64, 100, 128, 200]},
             scoring='accuracy')
```

In [30]:
```
grid.best_params_
```

Out[30]:
```
{'max_features': 2, 'n_estimators': 128}
```

In [31]:
```
predictions = grid.predict(X_test)
```

In [32]:
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

      Adelie       0.98      1.00      0.99        41
   Chinstrap       1.00      0.96      0.98        23
      Gentoo       1.00      1.00      1.00        37

    accuracy                           0.99       101
   macro avg       0.99      0.99      0.99       101
weighted avg       0.99      0.99      0.99       101
```

In [33]:
```
plot_confusion_matrix(grid,X_test,y_test)
```

Out[33]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x217c9c3d610
>
```