Multivariate Analysis

Classification

Predict the species of a penguin based on their characteristics.

In [1]:

```
import pandas as pd
import seaborn as sns
import numpy as np
```

In [2]:

```
penguins = sns.load_dataset("penguins")
```

In [3]:

```
penguins_no_na = penguins.dropna()
penguins_no_na.head()
```

Out[3]:

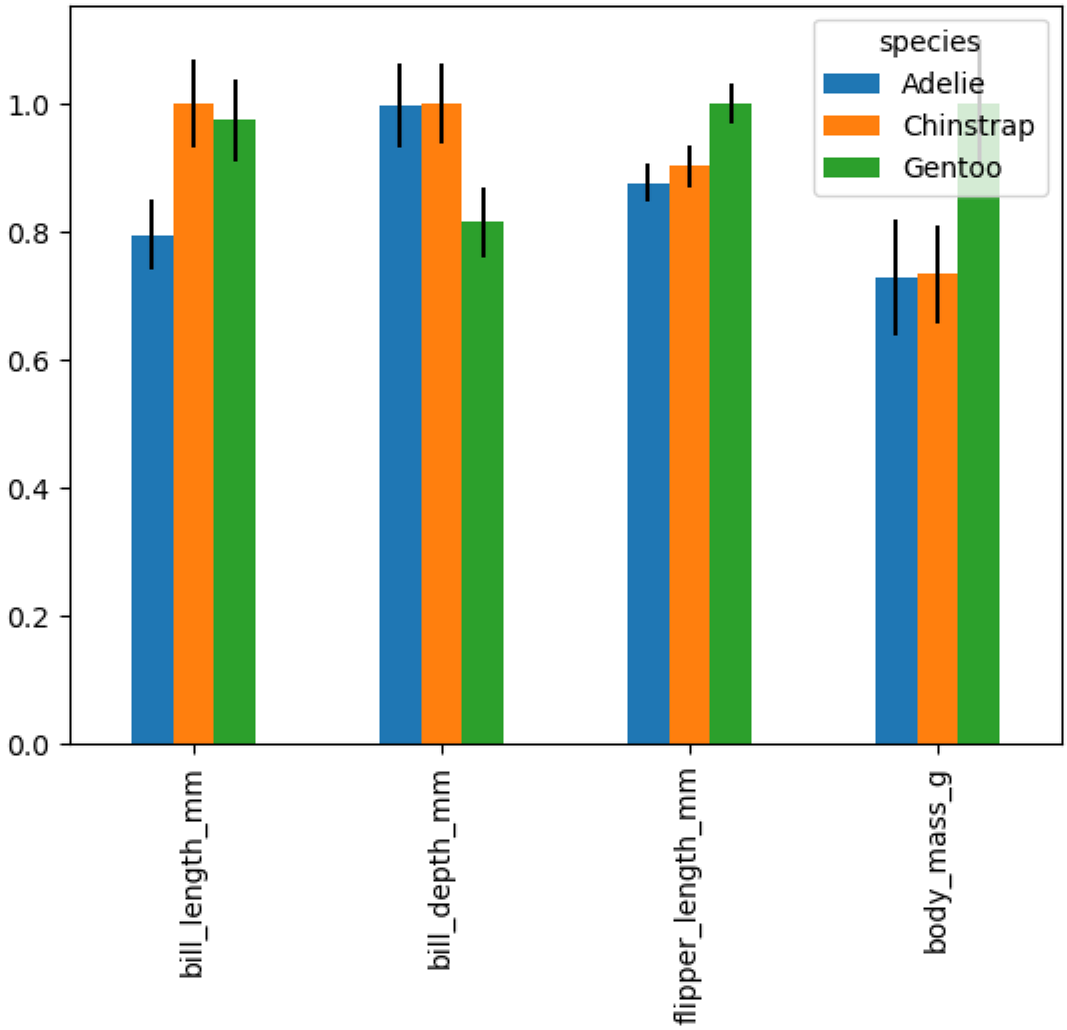
	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female
5	Adelie	Torgersen	39.3	20.6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

In [4]:

```
grouped = penguins_no_na.groupby('species')
normalized = (grouped.mean() / grouped.mean().max()).transpose()
errors = (grouped.std() / grouped.mean().max()).transpose()
normalized.plot(kind='bar', yerr=errors)
```

Out[4]:

```
<AxesSubplot:>
```



Op basis van bovenstaande grafiek is te verwachten dat species niet goed voorspeld kunnen worden als je maar één numerical gebruikt, er is altijd wel overlap tussen twee species' confidence intervals. Ook is het bij 3 van de 4 dezelfde species die overlap hebben, dus is het nodig om in ieder geval bill_length_mm toe te voegen, en ten minste één van de andere 3 numericals.

In [5]:
```
from sklearn.model_selection import train_test_split
```

In [6]:
```
penguins_train, penguins_test = train_test_split(penguins_no_na, test_size=0.3, random_s
```

In [7]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [8]:
```
features= ['bill_length_mm', 'flipper_length_mm','body_mass_g']
dt_classification = DecisionTreeClassifier(max_depth = 5)
dt_classification.fit(penguins_train[features], penguins_train['species'])
```

Out[8]:
```
DecisionTreeClassifier(max_depth=5)
```

Een diepte hoger dan 5 geeft niet meer nodes in de uiteindelijke tree.

In [9]:
```
def calculate_accuracy(predictions, actuals):
    if(len(predictions) != len(actuals)):
        raise Exception("The amount of predictions did not equal the amount of actuals")

    return (predictions == actuals).sum() / len(actuals)
```

In [10]:
```
predictionsOnTrainset = dt_classification.predict(penguins_train[features])
predictionsOnTestset = dt_classification.predict(penguins_test[features])
```

```
accuracyTrain = calculate_accuracy(predictionsOnTrainset, penguins_train.species)
accuracyTest = calculate_accuracy(predictionsOnTestset, penguins_test.species)

print("Accuracy on training set " + str(accuracyTrain))
print("Accuracy on test set " + str(accuracyTest))
```

```
Accuracy on training set 1.0
Accuracy on test set 0.93
```

De accuracy voor de test set is iets lager, wat niet heel raar is. Dit betekent dat het model niet perfect is, maar over het algemeen wel een vrij goede fit.

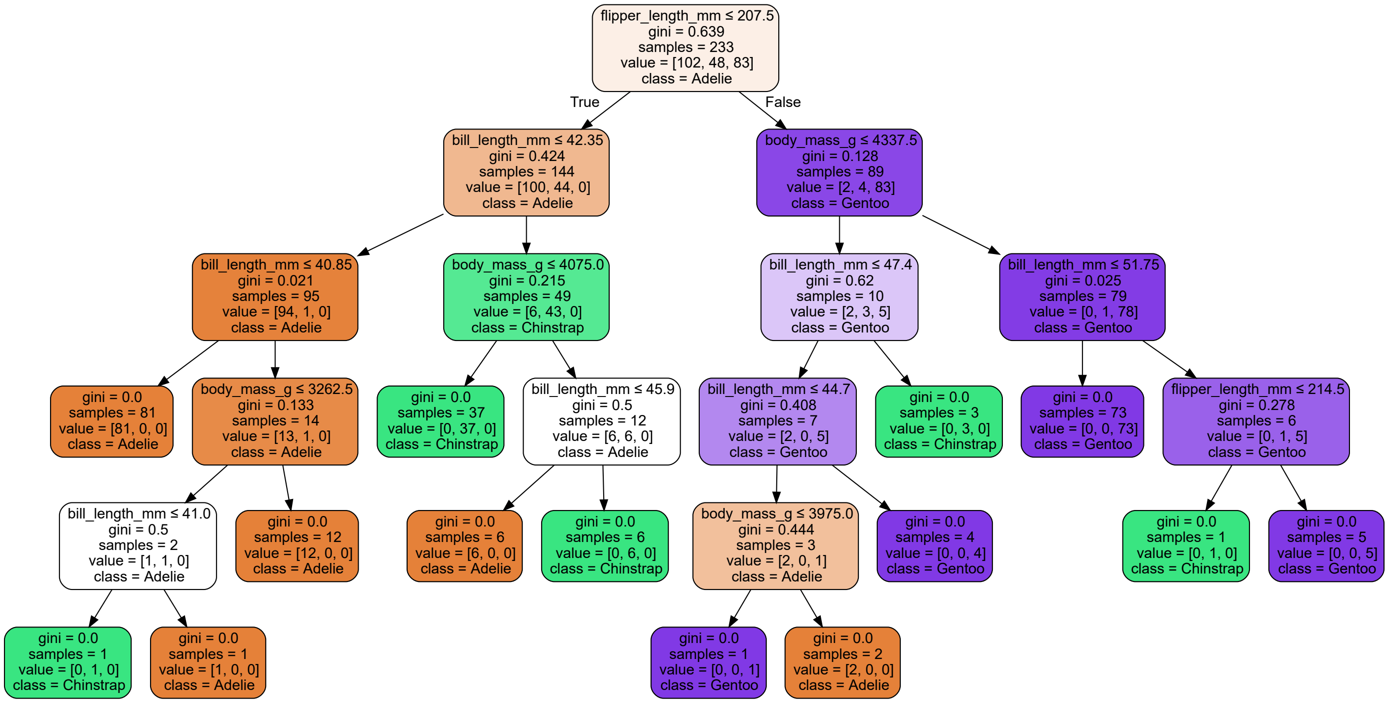
In [11]:

```
from sklearn import tree
import graphviz

def plot_tree_classification(model, features, class_names):
    # Generate plot data
    dot_data = tree.export_graphviz(model, out_file=None,
                                    feature_names=features,
                                    class_names=class_names,
                                    filled=True, rounded=True,
                                    special_characters=True)

    # Turn into graph using graphviz
    graph = graphviz.Source(dot_data)

    # Write out a pdf
    graph.render("Trees/decision_tree_15")

    # Display in the notebook
    return graph
```

In [12]:

```
plot_tree_classification(dt_classification, features, np.sort(penguins.species.unique()))
```

Out[12]:


flipper_length_mm ≤ 207.5
gini = 0.639
samples = 233
value = [102, 48, 83]
class = Adelie
True
False
bill_length_mm ≤ 42.35
gini = 0.424
samples = 144
value = [100, 44, 0]
class = Adelie
body_mass_g ≤ 4337.5
gini = 0.128
samples = 89
value = [2, 4, 83]
class = Gentoo
bill_length_mm ≤ 40.85
gini = 0.021
samples = 95
value = [94, 1, 0]
class = Adelie
body_mass_g ≤ 4075.0
gini = 0.215
samples = 49
value = [6, 43, 0]
class = Chinstrap
bill_length_mm ≤ 47.4
gini = 0.62
samples = 10
value = [2, 3, 5]
class = Gentoo
bill_length_mm ≤ 51.75
gini = 0.025
samples = 79
value = [0, 1, 78]
class = Gentoo
gini = 0.0
samples = 81
value = [81, 0, 0]
class = Adelie
body_mass_g ≤ 3262.5
gini = 0.133
samples = 14
value = [13, 1, 0]
class = Adelie
gini = 0.0
samples = 37
value = [0, 37, 0]
class = Chinstrap
bill_length_mm ≤ 45.9
gini = 0.5
samples = 12
value = [6, 6, 0]
class = Adelie
bill_length_mm ≤ 44.7
gini = 0.408
samples = 7
value = [2, 0, 5]
class = Gentoo
gini = 0.0
samples = 3
value = [0, 3, 0]
class = Chinstrap
gini = 0.0
samples = 73
value = [0, 0, 73]
class = Gentoo
flipper_length_mm ≤ 214.5
gini = 0.278
samples = 6
value = [0, 1, 5]
class = Gentoo
bill_length_mm ≤ 41.0
gini = 0.5
samples = 2
value = [1, 1, 0]
class = Adelie
gini = 0.0
samples = 12
value = [12, 0, 0]
class = Adelie
gini = 0.0
samples = 6
value = [6, 0, 0]
class = Adelie
gini = 0.0
samples = 6
value = [0, 6, 0]
class = Chinstrap
body_mass_g ≤ 3975.0
gini = 0.444
samples = 3
value = [2, 0, 1]
class = Adelie
gini = 0.0
samples = 4
value = [0, 0, 4]
class = Gentoo
gini = 0.0
samples = 1
value = [0, 1, 0]
class = Chinstrap
gini = 0.0
samples = 5
value = [0, 0, 5]
class = Gentoo
gini = 0.0
samples = 1
value = [0, 1, 0]
class = Chinstrap
gini = 0.0
samples = 1
value = [1, 0, 0]
class = Adelie
gini = 0.0
samples = 1
value = [0, 0, 1]
class = Gentoo
gini = 0.0
samples = 2
value = [2, 0, 0]
class = Adelie