In [3]:
```
import pandas as pd
import seaborn as sns
penguins = sns.load_dataset("penguins")
```

In [4]:
```
penguins.head()
```

Out[4]:

	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
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female

In [5]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [6]:
```
features= ['bill_length_mm']
dt = DecisionTreeClassifier(max_depth = 2) # Increase max_depth to see effect in the plot
```

In [7]:
```
penguinsnansense = penguins.dropna()
```

In [8]:
```
dt.fit(penguinsnansense[features], penguinsnansense['species'])
```

Out[8]:
```
DecisionTreeClassifier(max_depth=2)
```

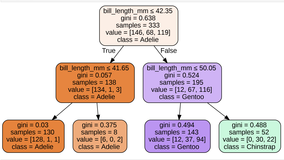
In [9]:
```
conda install -c anaconda python-graphviz
```

```
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

# All requested packages already installed.


Note: you may need to restart the kernel to use updated packages.
```

In [10]:
```
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("decision_tree")

    # Display in the notebook
    return graph
```

In [11]:
```
plot_tree_classification(dt, features, penguinsnansense.species.unique())
```

Out[11]:



In [12]:
```
predictions = dt.predict(penguinsnansense[features])
```

In [13]:
```
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 [14]:
```
calculate_accuracy(predictions, penguinsnansense.species)
```

Out[14]:
```
0.7747747747747747
```

In [15]:
```
from sklearn.model_selection import train_test_split
```

In [16]:
```
penguins_train, penguins_test = train_test_split(penguinsnansense, test_size=0.3, stratify=penguinsnansense['species'], random_state=42)
print(penguins_train.shape, penguins_test.shape)
```

```
(233, 7) (100, 7)
```

In [17]:
```
features= ['bill_length_mm']
dt_classification = DecisionTreeClassifier(max_depth = 1) # Increase max_depth to see effect in the plot
dt_classification.fit(penguins_train[features], penguins_train['species'])
```

Out[17]:
```
DecisionTreeClassifier(max_depth=1)
```

In [18]:
```
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 0.7596566523605151
Accuracy on test set 0.73
```

Accuracy is different, this was expected since one of the 2 groups is bigger than the other. More data is more accuracy.

In [19]:
```
penguins_train
```

Out[19]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
268	Gentoo	Biscoe	44.9	13.3	213.0	5100.0	Female
201	Chinstrap	Dream	49.8	17.3	198.0	3675.0	Female
321	Gentoo	Biscoe	55.9	17.0	228.0	5600.0	Male
341	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	Male
13	Adelie	Torgersen	38.6	21.2	191.0	3800.0	Male
…	…	…	…	…	…	…	…
14	Adelie	Torgersen	34.6	21.1	198.0	4400.0	Male
102	Adelie	Biscoe	37.7	16.0	183.0	3075.0	Female
337	Gentoo	Biscoe	48.8	16.2	222.0	6000.0	Male
306	Gentoo	Biscoe	43.4	14.4	218.0	4600.0	Female
70	Adelie	Torgersen	33.5	19.0	190.0	3600.0	Female

233 rows × 7 columns

In [21]:
```
penguins_test
```

Out[21]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
194	Chinstrap	Dream	50.9	19.1	196.0	3550.0	Male
235	Gentoo	Biscoe	49.3	15.7	217.0	5850.0	Male
289	Gentoo	Biscoe	50.7	15.0	223.0	5550.0	Male
308	Gentoo	Biscoe	47.5	14.0	212.0	4875.0	Female
61	Adelie	Torgersen	42.9	17.6	196.0	4700.0	Male
…	…	…	…	…	…	…	…
274	Gentoo	Biscoe	46.5	14.4	217.0	4900.0	Female
214	Chinstrap	Dream	45.7	17.0	195.0	3650.0	Female
146	Adelie	Dream	39.2	18.6	190.0	4250.0	Male
65	Adelie	Biscoe	41.6	18.0	192.0	3950.0	Male
318	Gentoo	Biscoe	48.4	14.4	203.0	4625.0	Female

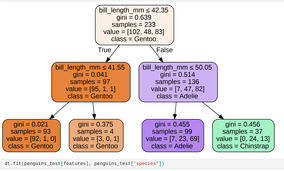
100 rows × 7 columns

In [22]:
```
dt.fit(penguins_train[features], penguins_train['species'])
```

Out[22]:
```
DecisionTreeClassifier(max_depth=2)
```

In [23]:
```
plot_tree_classification(dt, features, penguins_train.species.unique())
```

Out[23]:



In [24]:
```
dt.fit(penguins_test[features], penguins_test['species'])
```

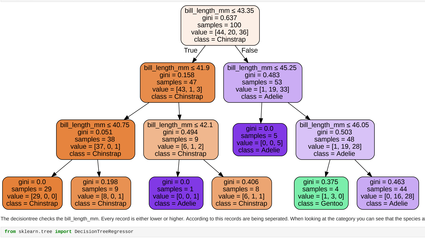
Out[24]:
```
DecisionTreeClassifier(max_depth=2)
```

In [25]:
```
features= ['bill_length_mm']
dt = DecisionTreeClassifier(max_depth = 3) # Increase max_depth to see effect in the plot
dt.fit(penguins_test[features], penguins_test['species'])
```

Out[25]:
```
DecisionTreeClassifier(max_depth=3)
```

In [26]:
```
plot_tree_classification(dt, features, penguins_test.species.unique())
```

Out[26]:



The decisionmree checks the bill_length_mm. Every record is either lower or higher. According to this records are being seperated. When looking at the category you can see that the species affects the bill_length_mm of penguins

In [30]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [31]:
```
penguins.corr()
```

Out[31]:

	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
bill_length_mm	1.000000	-0.235053	0.656181	0.595110
bill_depth_mm	-0.235053	1.000000	-0.583851	-0.471916
flipper_length_mm	0.656181	-0.583851	1.000000	0.871202
body_mass_g	0.595110	-0.471916	0.871202	1.000000

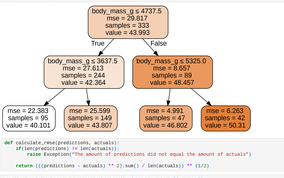
In [43]:
```
features= ['body_mass_g']
dt_regression = DecisionTreeRegressor(max_depth = 2) # Increase max_depth to see effect in the plot
dt_regression.fit(penguinsnansense[features], penguinsnansense['bill_length_mm'])
```

Out[43]:
```
DecisionTreeRegressor(max_depth=2)
```

In [44]:
```
from sklearn import tree
import graphviz

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

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

    # Write out a pdf
    graph.render("decision_tree")

    # Display in the notebook
    return graph
```

In [45]:
```
plot_tree_regression(dt_regression, features)
```

Out[45]:



In [38]:
```
def calculate_rmse(predictions, actuals):
    if(len(predictions) != len(actuals)):
        raise Exception("The amount of predictions did not equal the amount of actuals")

    return ((predictions - actuals) ** 2).sum() / len(actuals)) ** (1/2)
```

In [39]:
```
predictionsOnTrainset = dt_regression.predict(penguins_train[features])
predictionsOnTestset = dt_regression.predict(penguins_test[features])

rmseTrain = calculate_rmse(predictionsOnTrainset, penguins_train.bill_length_mm)
rmseTest = calculate_rmse(predictionsOnTestset, penguins_test.bill_length_mm)

print('RMSE on training set " + str(rmseTrain))
print('RMSE on test set " + str(rmseTest))
```

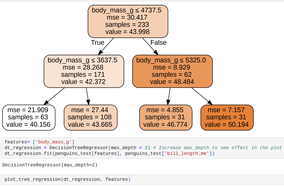
```
RMSE on training set 4.499031806368162
RMSE on test set 4.154335247607651
```

In [46]:
```
features= ['body_mass_g']
dt_regression = DecisionTreeRegressor(max_depth = 2) # Increase max_depth to see effect in the plot
dt_regression.fit(penguins_train[features], penguins_train['bill_length_mm'])
```

Out[46]:
```
DecisionTreeRegressor(max_depth=2)
```

In [47]:
```
plot_tree_regression(dt_regression, features)
```

Out[47]:



In [48]:
```
features= ['body_mass_g']
dt_regression = DecisionTreeRegressor(max_depth = 2) # Increase max_depth to see effect in the plot
dt_regression.fit(penguins_test[features], penguins_test['bill_length_mm'])
```

Out[48]:
```
DecisionTreeRegressor(max_depth=2)
```

In [49]:
```
plot_tree_regression(dt_regression, features)
```

Out[49]:

The decisionmree checks the body_mass_g. Every record is either lower or higher. According to this records are being seperated. You will get a clearer understanding of which bill_length_mm value correlates with which body_mass_g value

In [ ]:
```
```