Multivariate Analysis

Regression

Predicting the body mass of a penguin based on other characteristics

In [13]:
```
import pandas as pd
import seaborn as sns
import numpy as np
```

In [14]:
```
penguins = sns.load_dataset("penguins")
penguins_no_na = penguins.dropna()
penguins_no_na.head()
```

Out[14]:

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

In [16]:
```
penguins_train, penguins_test = train_test_split(penguins_no_na, test_size=0.3,  random_
```

In [17]:
```
penguins.corr().style.background_gradient(cmap='coolwarm', axis=None).format(precision=2
```

Out[17]:

	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
bill_length_mm	1.00	-0.24	0.66	0.60
bill_depth_mm	-0.24	1.00	-0.58	-0.47
flipper_length_mm	0.66	-0.58	1.00	0.87
body_mass_g	0.60	-0.47	0.87	1.00

Body mass heeft vooral correlatie met flipper length, maar ook wat met bill length en depth, dus we kiezen alle 3 de kolommen om mee te voorspellen.

In [18]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [19]:
```
features= ['flipper_length_mm', 'bill_length_mm', 'bill_depth_mm']
dt_regression = DecisionTreeRegressor(max_depth = 3)
dt_regression.fit(penguins_train[features], penguins_train['body_mass_g'])
```

Out[19]:
```
DecisionTreeRegressor(max_depth=3)
```

In [20]:
```
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 [21]:
```
predictionsOnTrainset = dt_regression.predict(penguins_train[features])
predictionsOnTestset = dt_regression.predict(penguins_test[features])

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

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

```
RMSE on training set 335.37272629423245
RMSE on test set 350.41154581335104
```

RMSE op zichzelf zegt niet heel erg veel, omdat het erg afhangt van de grootte van de waarden van die kolom, of het een goede fout is of niet.

In [22]:
```
print("Normalised RMSE on training set " + str(rmseTrain/ penguins_train.body_mass_g.std
print("Normalised RMSE on test set " + str(rmseTest/ penguins_test.body_mass_g.std()))
```

```
Normalised RMSE on training set 0.41029685555550044
Normalised RMSE on test set 0.4493378306619543
```

De RMSE is niet erg laag, dus waarschijnlijk heeft de decision tree regressor geen goede fit kunnen vinden voor de body mass.

Hoewel de RMSE voor de training set steeds kleiner wordt met een grotere diepte, is de RMSE van de test set het kleinst bij een diepte van 3.

In [23]:
```
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("Trees/decision_tree_17")

    # Display in the notebook
    return graph
```

In [24]:
```
plot_tree_regression(dt_regression, features)
```

Out[24]: