Portfolio assignment 17

30 min: Train a decision tree to predict the body_mass_g of a penguin based on their characteristics.

- Split the penguin dataset into a train (70%) and test (30%) set.
- Use the train set to fit a DecisionTreeRegressor. You are free to to choose which columns you want to use as feature variables and you are also free to choose the max_depth of the tree. **Note**: Some machine learning algorithms can not handle missing values. You will either need to
  - replace missing values (with the mean or most popular value). For replacing missing values you can use .fillna(<value>) https://pandas.pydata.org/docs/reference/api/pandas.Series.fillna.html (https://pandas.pydata.org/docs/reference/api/pandas.Series.fillna.html)
  - remove rows with missing data. You can remove rows with missing data with .dropna() https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.dropna.html (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.dropna.html)
- Use your decision tree model to make predictions for both the train and test set.
- Calculate the RMSE for both the train set predictions and test set predictions.
- Is the RMSE different? Did you expect this difference?
- Use the plot_tree_regression function above to create a plot of the decision tree. Take a few minutes to analyse the decision tree. Do you understand the tree?

In [1]:

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
import seaborn as sns
```

In [2]:

```
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 [3]:

```
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)

## The same function but using a for-loop instead of a vectorized operation.
# def calculate_rmse(predictions, actuals):
#     if(len(predictions) != len(actuals)):
#         raise Exception("The amount of predictions did not equal the amount of actuals")
#
#     diffSquared = 0
#
#     for prediction_i, actual_i in zip(predictions, actuals):
#         diffSquared += (prediction_i - actual_i)**2
#
#     return (diffSquared/len(actuals))**(1/2)
```

In [4]:

```
penguins = sns.load_dataset("penguins")
penguins.dropna(axis=0, inplace= True)
```

In [5]:

```
penguins_train, penguins_test = train_test_split(penguins, test_size = 0.3, stratify=pengui
print(penguins_train.shape, penguins_test.shape)
```

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

In [6]:

```
features= ['body_mass_g']
dt_regression = DecisionTreeRegressor(max_depth = 3) # Increase max_depth to see effect in
dt_regression.fit(penguins_train[features], penguins_train['flipper_length_mm'])
```

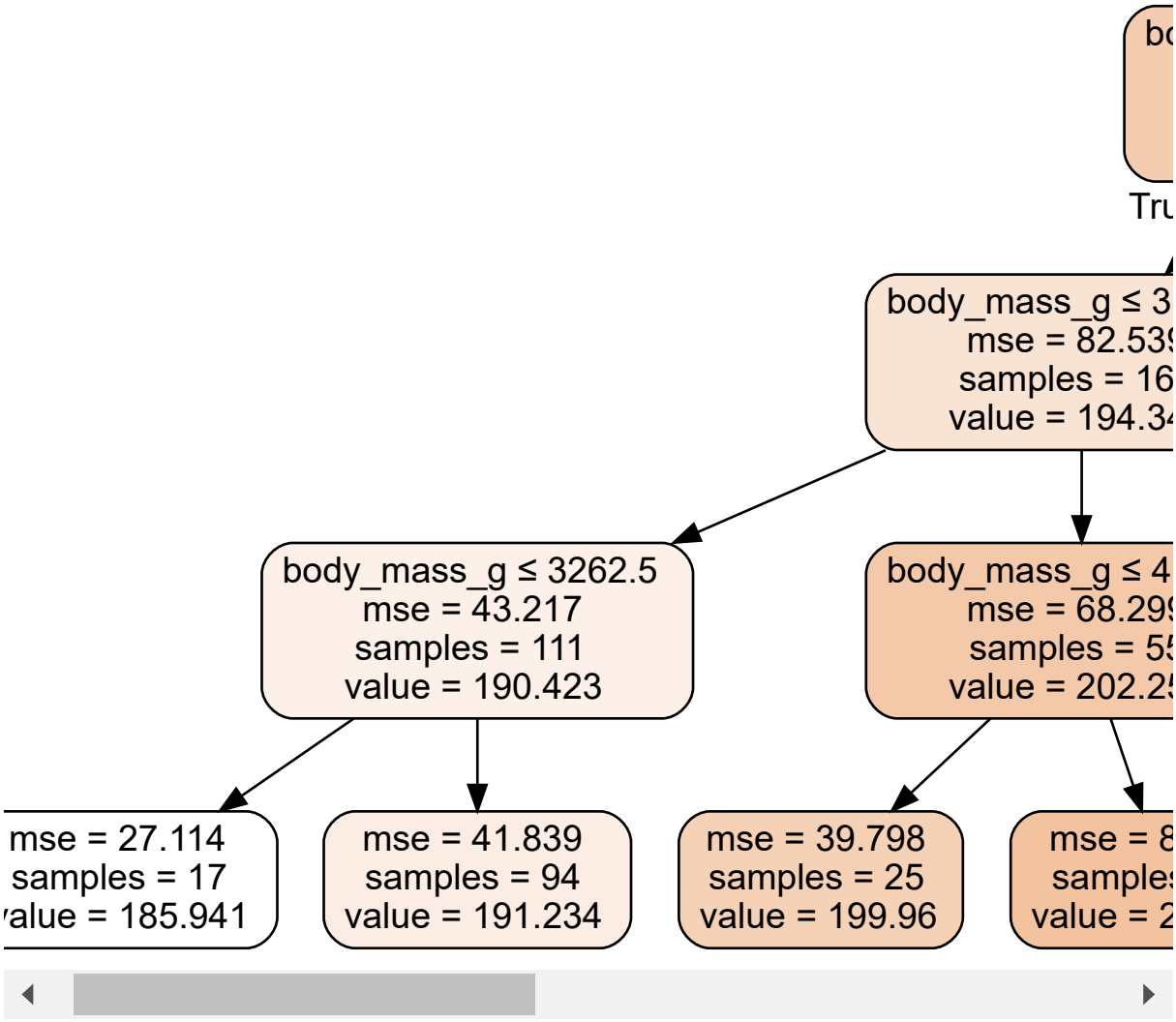
Out[6]:

```
DecisionTreeRegressor(max_depth=3)
```

In [7]:

```
plot_tree_regression(dt_regression, features)
```

Out[7]:



You can definetly see that the body mass can influence on other attributes of the penguin.