Portfolio assignment 18

30 min: Train a decision tree to predict one of the numerical columns of your own dataset.

- Split your 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.
- 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 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]:
```
import pandas as pd
import seaborn as sns
```

I would love to use my current dataset, but there's not really 2 numerical values that would give a nice result.

In [2]:
```
students = pd.read_csv('StudentsPerformance.csv')
students.head()
```

Out[2]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

I feel like you could get a proper result from this dataset.

In [3]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [4]:
```
features= ['math score']
dt_regression = DecisionTreeRegressor(max_depth = 3)  # Increase max_depth to see effe
dt_regression.fit(students[features], students['reading score'])
```

Out[4]:
```
DecisionTreeRegressor(max_depth=3)
```

In [5]:
```
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 [6]:
```
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 [7]:
```
predictionsOnTrainset = dt_regression.predict(students[features])
predictionsOnTestset = dt_regression.predict(students[features])

rmseTrain = calculate_rmse(predictionsOnTrainset, students['reading score'])
rmseTest = calculate_rmse(predictionsOnTestset, students['reading score'])

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

```
RMSE on training set 8.443068408025608
RMSE on test set 8.443068408025608
```

Both values are quite low, but the difference between the two values is almost zero. At least it means both sets are very equal to each other.

In [8]:
```
plot_tree_regression(dt_regression, features)
```

Out[8]: