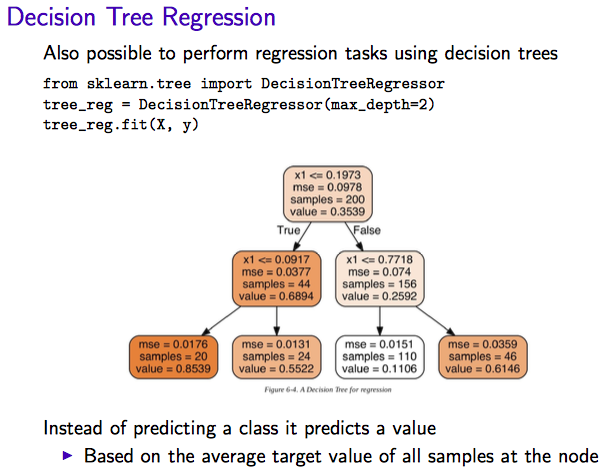
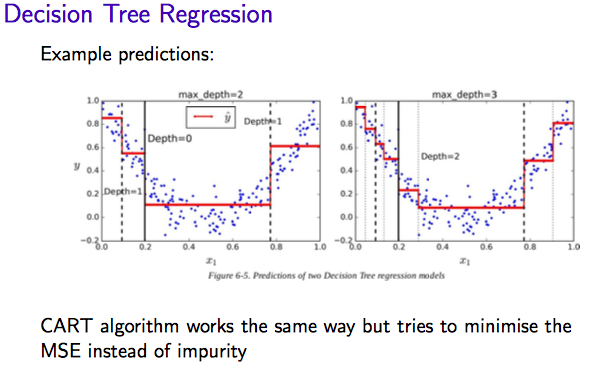
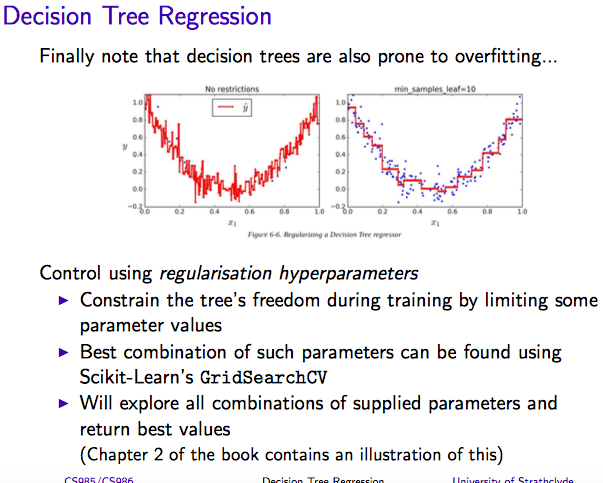
Fundamentals of Machine Learning- Group Project

Decision Tree Regression:







**Decision Tree Regression**

from sklearn.tree import DecisionTreeRegressor

bX, by = datasets.load\_boston(return\_X\_y=True)

bXtrain, bXtest, bytrain, bytest = train\_test\_split(bX, by, test\_size=0.2, random\_state=42)

tree\_reg = DecisionTreeRegressor(min\_samples\_leaf=3) # explore min\_samples = e.g. put to 1 to see what happens

tree\_reg.fit(bXtrain,bytrain)

boston\_tree\_train\_preds = tree\_reg.predict(bXtrain) # Check fit

lin\_mse = mean\_squared\_error(bytrain, boston\_tree\_train\_preds)

lin\_rmse = np.sqrt(lin\_mse)

lin\_rmse

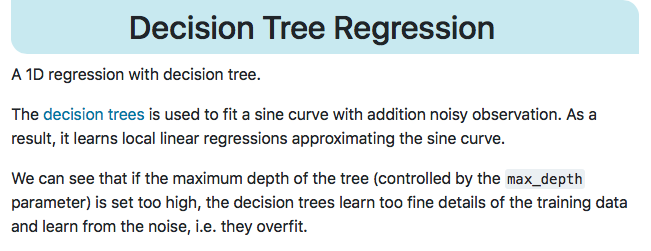
# Check on test set

boston\_tree\_test\_preds = tree\_reg.predict(bXtest)

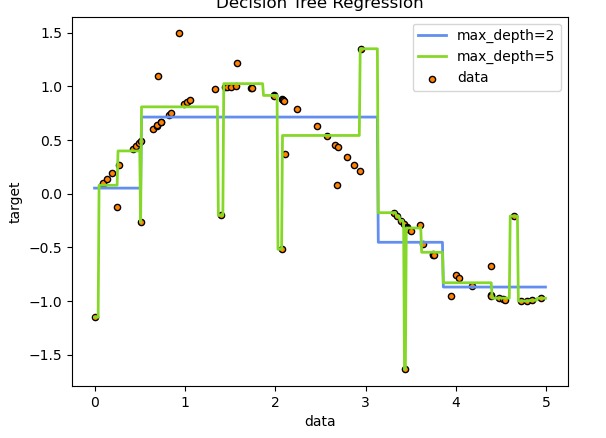
lin\_mse = mean\_squared\_error(bytest, boston\_tree\_test\_preds)

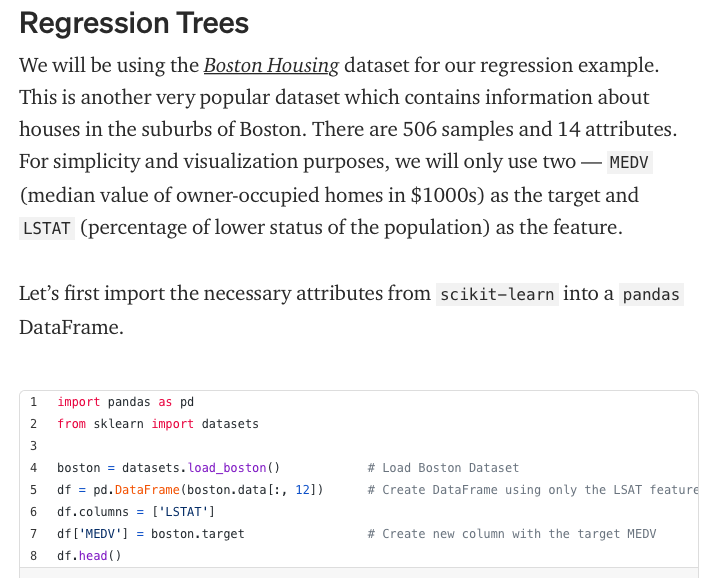
lin\_rmse = np.sqrt(lin\_mse)

lin\_rmse

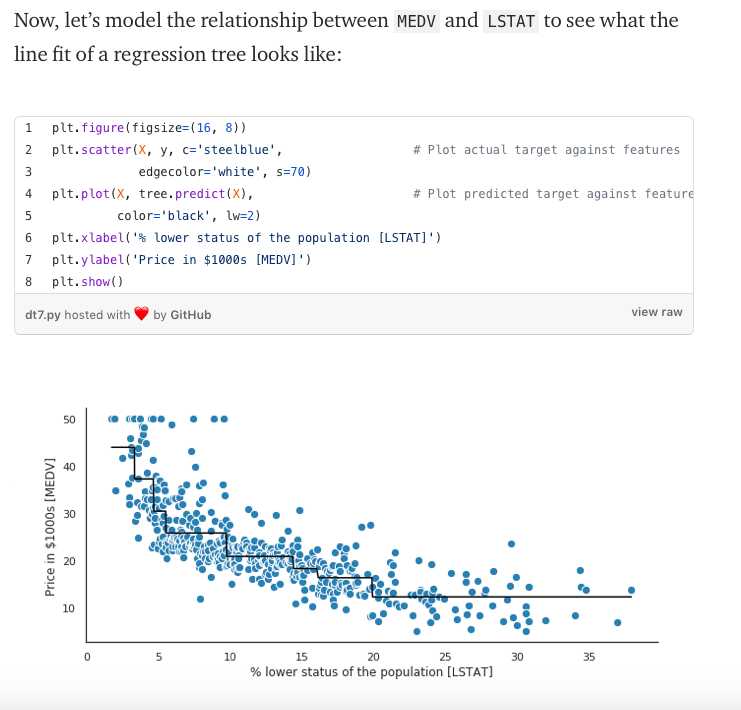












[**https://gdcoder.com/decision-tree-regressor-explained-in-depth/**](https://gdcoder.com/decision-tree-regressor-explained-in-depth/)

### How it makes predictions?

Given a data point you run it through the entirely tree asking True/False questions up until it reaches a leaf node. The final prediction is the average of the value of the dependent variable in that leaf node.

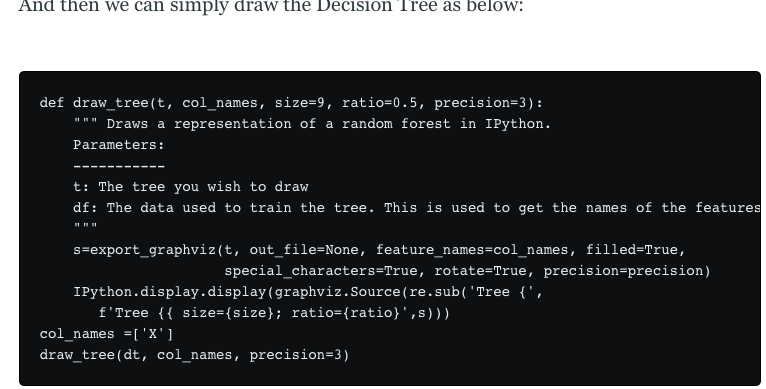
**Use Random Forest**

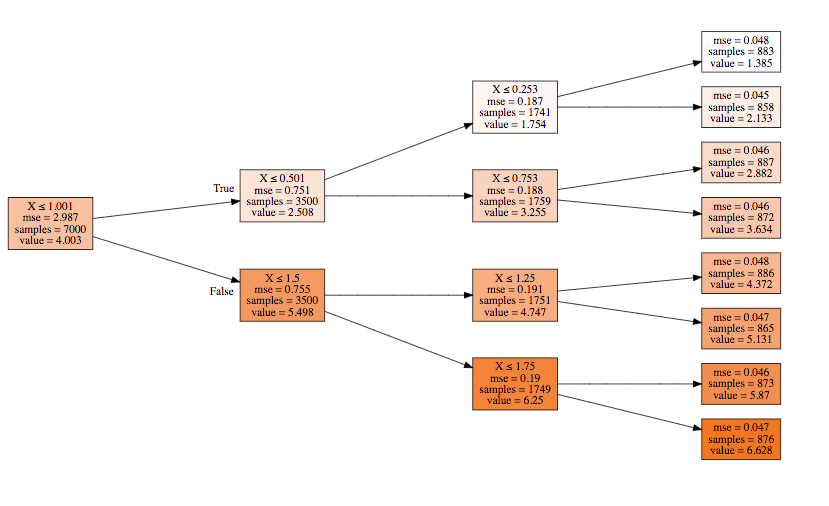
It seems that not many people actually take the time to prune a decision tree or fine tuning but rather they select to use a random forest regressor (a collection of decision trees) which are less prone to overfitting and perform better than a single optimised tree.

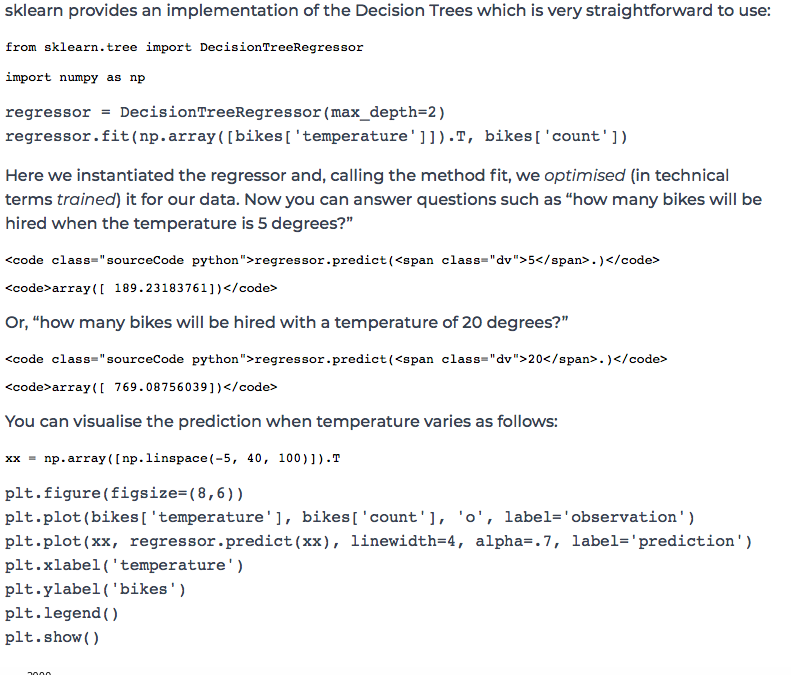
### 

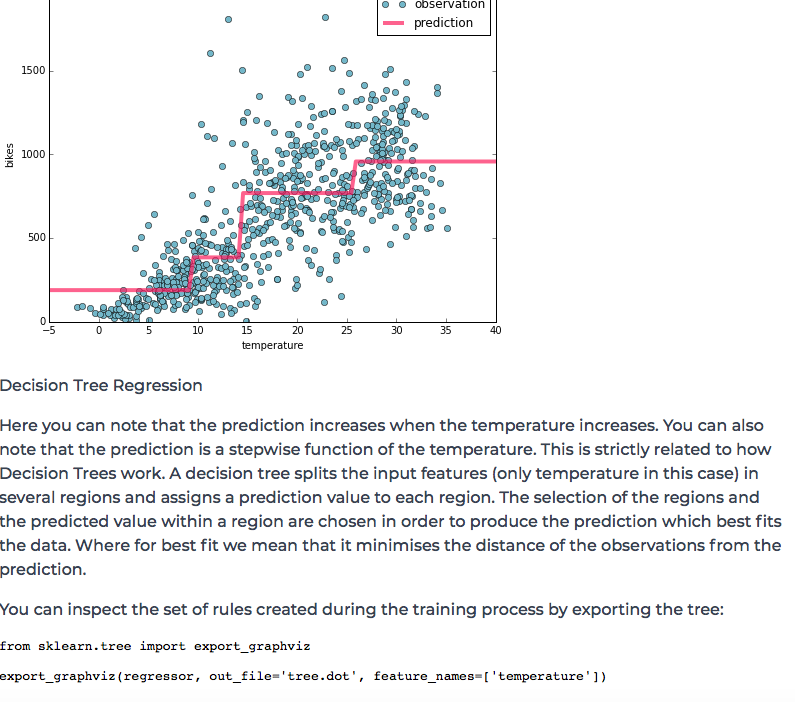
## Visualise a Decision Tree model

### 

****

****

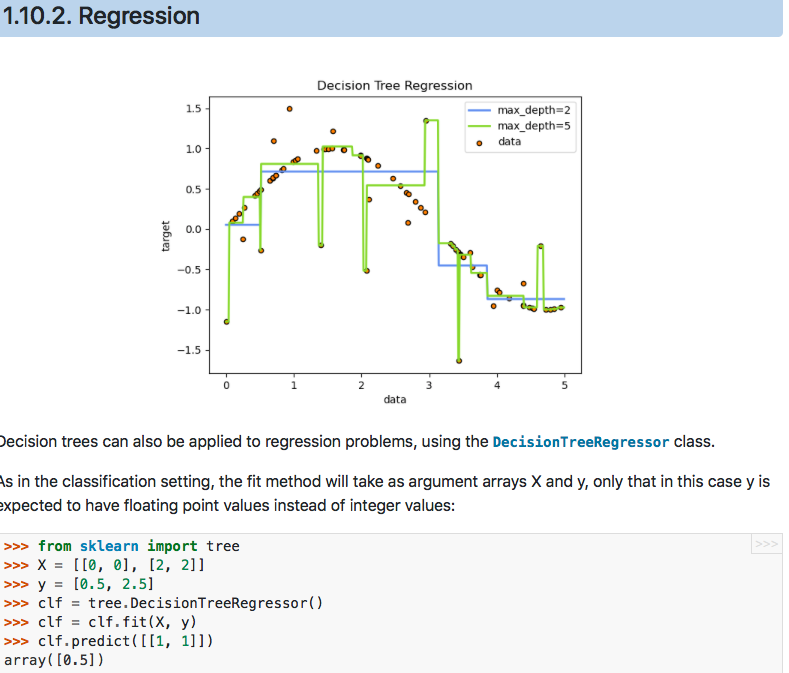
****

****

[**https://levelup.gitconnected.com/decision-tree-regression-df9e24ffe59a**](https://levelup.gitconnected.com/decision-tree-regression-df9e24ffe59a)

[**https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html**](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html)

[**https://christophm.github.io/interpretable-ml-book/tree.html**](https://christophm.github.io/interpretable-ml-book/tree.html)

****

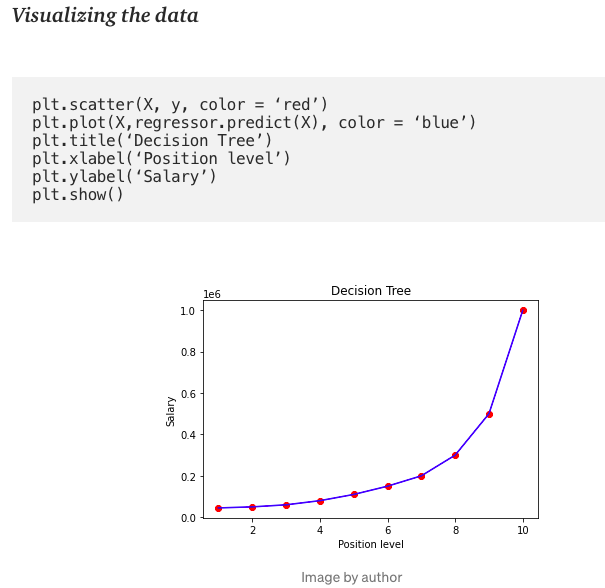
[**https://www.geeksforgeeks.org/python-decision-tree-regression-using-sklearn/**](https://www.geeksforgeeks.org/python-decision-tree-regression-using-sklearn/)

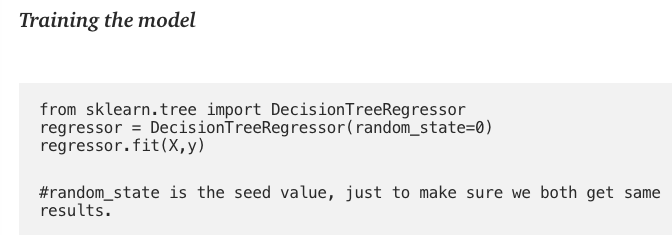
[**https://stackabuse.com/decision-trees-in-python-with-scikit-learn/**](https://stackabuse.com/decision-trees-in-python-with-scikit-learn/)

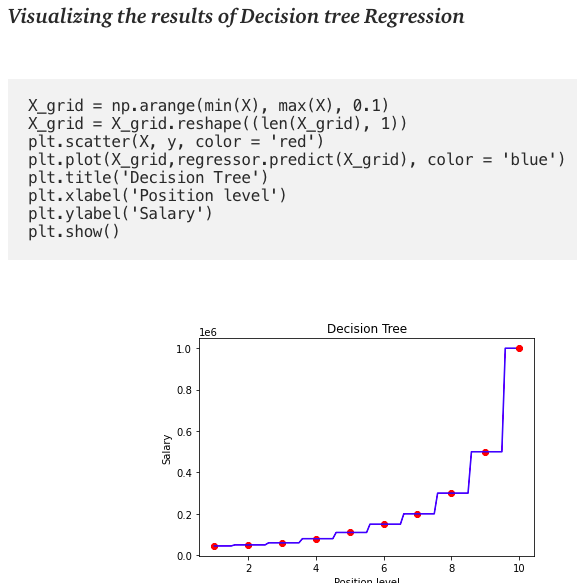
[**https://medium.com/pursuitnotes/decision-tree-regression-in-6-steps-with-python-1a1c5aa2ee16**](https://medium.com/pursuitnotes/decision-tree-regression-in-6-steps-with-python-1a1c5aa2ee16)

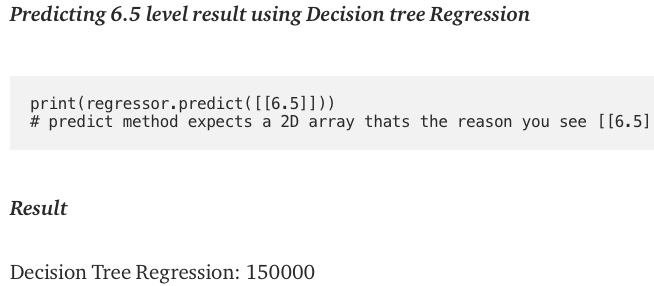
**Baby Steps::**

****

****

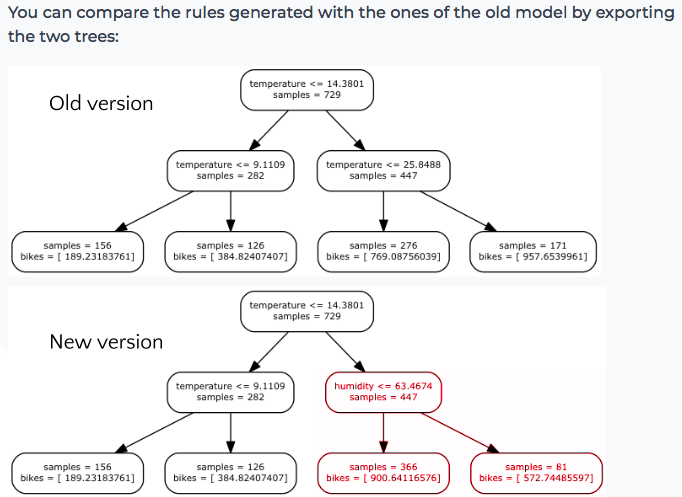
****

****

****

**From simple regression to multiple regression with decision trees:**





[**https://heartbeat.fritz.ai/implementing-regression-using-a-decision-tree-and-scikit-learn-ac98552b43d7**](https://heartbeat.fritz.ai/implementing-regression-using-a-decision-tree-and-scikit-learn-ac98552b43d7)

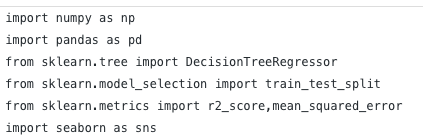
Below are the cases where you would likely prefer a decision tree over other regression algorithms:

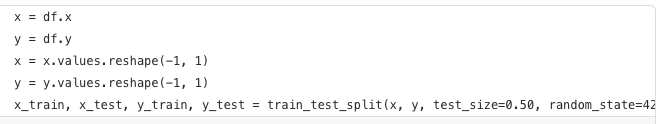
1. There are non-linear or complex relationships between features and labels
2. You need a model that is easy to explain

# **Decision trees: Key terms**

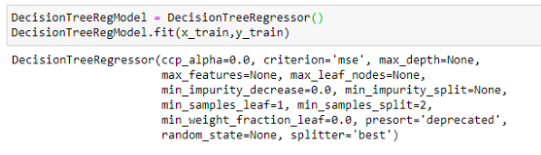
* **Root Node:** The top-most decision node in a decision tree.
* **Decision Node:** A tree node or parent node that splits into one ore more child nodes is called a decision node.
* **Leaf or Terminal Node:**Bottom nodes that (generally speaking) don’t split any further.
* **Splitting**: Process of dividing a node into two or more child nodes.
* **Pruning**: The opposite process of splitting. Removing the child nodes of a decision node is called pruning.

**Import the libraries**



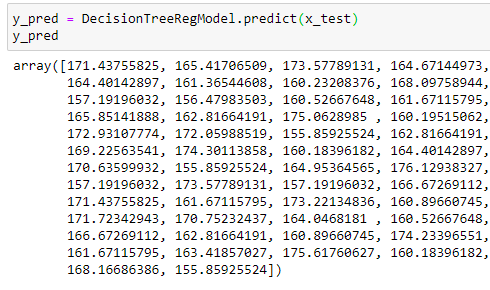


# **Fit the model:**



**Predict using the trained model**

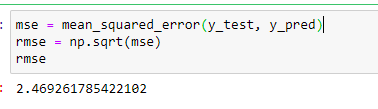
Once the model is trained, it’s ready to make predictions. We can use the predict method on the model and pass x\_test as a parameter to get the output as y\_pred.Notice that the prediction output is an array of real numbers corresponding to the input array.



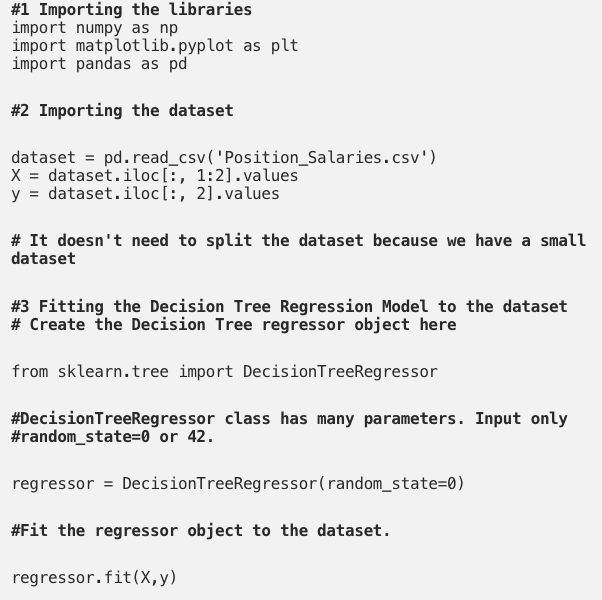
**Model evaluation**

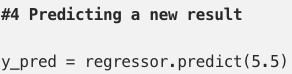
Finally, we need to check to see how well our model is performing on the test data. For this, we evaluate our model by finding the root mean squared error produced by the model.

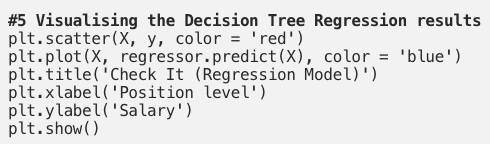
Mean squared error is a built in function, and we are using NumPy’s square root function (np.sqrt) on top of it to find the root mean squared error value.



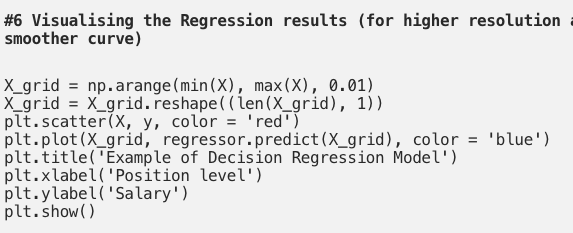
**https://medium.com/pursuitnotes/decision-tree-regression-in-6-steps-with-python-1a1c5aa2ee16**

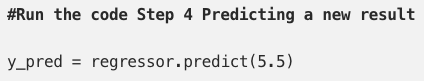






The Decision Tree Regression is both non-linear and non-continuous model so that the graph above seems problematic. So, I named it as “Check It” graph. If we code for higher resolution and smooth curve, it seems like below.





[**https://datascience.foundation/sciencewhitepaper/understanding-decision-trees-with-python**](https://datascience.foundation/sciencewhitepaper/understanding-decision-trees-with-python)

Decision Trees are popular because they have two key properties:

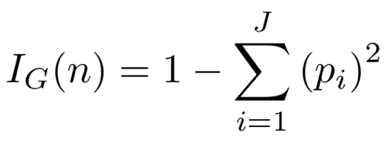
1. Simplicity: Decision Trees are simple, visually appealing and are easy to interpret.
2. Accuracy: Advance Decision Tree models show exceptional performance in predicting patterns in complex data.

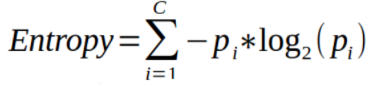
#### **Regression Tree**

Regression Trees are used for continuous quantitative target variables.  
Example: Predicting rainfall; Predicting revenue; Predicting marks etc.

**Gini Impurity –** As per Wikipedia, Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.

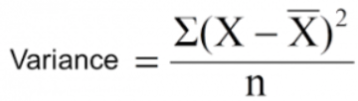
In simple terms, Gini impurity is the measure of impurity in a node. Its formula is:

**  
Entropy –**Another very popular way to split nodes in the decision tree is Entropy. Entropy is the measure of Randomness in the system. The formula for Entropy is:

****

**Variance –**– Gini Impurity and Entropy work well for the classification scenario.  
But what about regression?

In the case of regression, the most common split measure used is just the weighted variance of the nodes. It makes sense too: We want minimum variation in the nodes after the split.

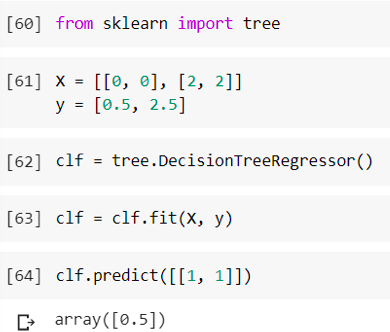
****

#### **Regression**

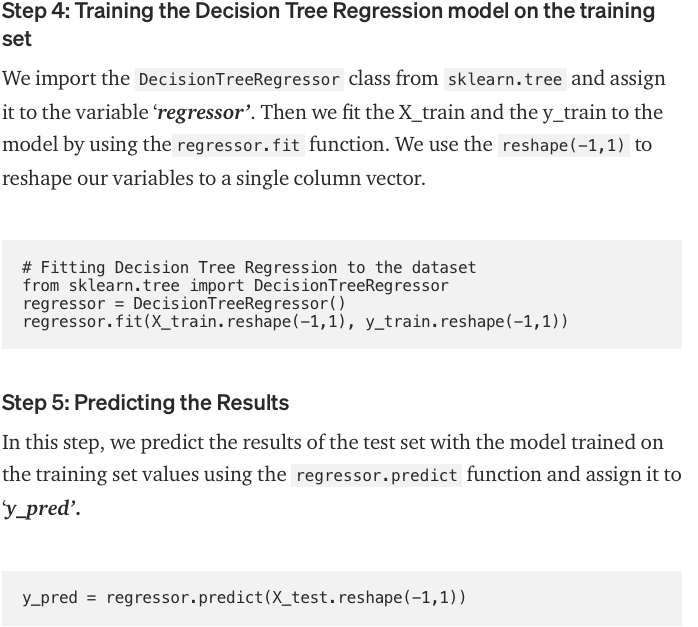
Decision trees can also be applied to regression problems, using the **DecisionTreeRegressor class.**

As in the classification setting, the fit method will take argument arrays X and y, only in this case y is expected to have floating point values instead of integer values.

Similar to what we did for **DecisionTreeClassifier,** in same way we will be using **DecisionTreeRegressor** method.

****

**https://towardsdatascience.com/machine-learning-basics-decision-tree-regression-1d73ea003fda**

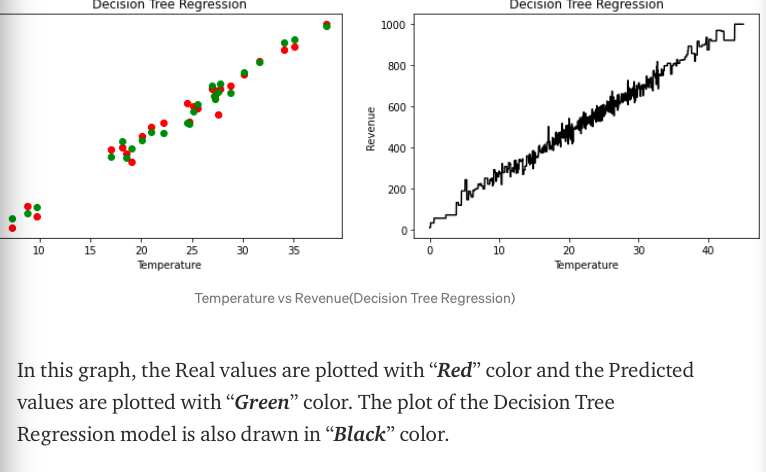
****

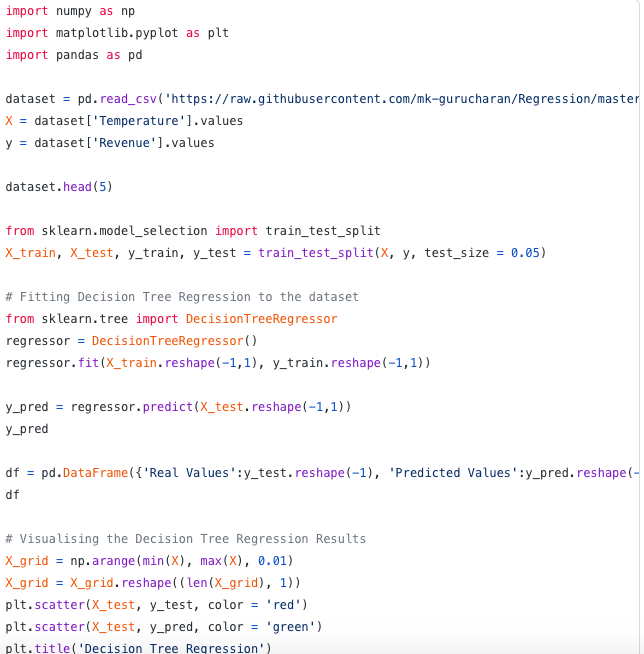
## **Step 6: Comparing the Real Values with Predicted Values**

In this step, we shall compare and display the values of y\_test as ‘**Real Values**’ and y\_pred as ‘**Predicted Values**’ in a Pandas dataframe.

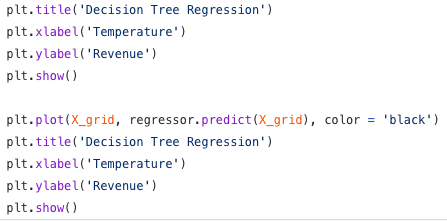


## **Step 7: Visualising the Decision Tree Regression Results**

****

****

****

****