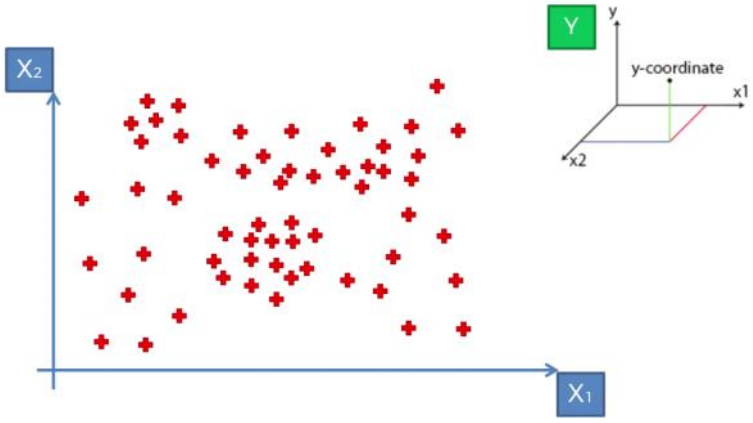
Decision Tree - Regression

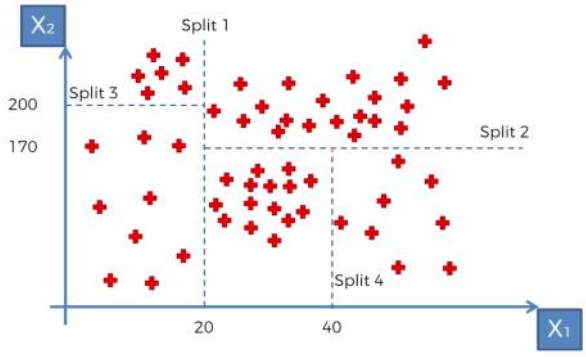
Thursday, January 19, 2017 6:43 AM

Intuition

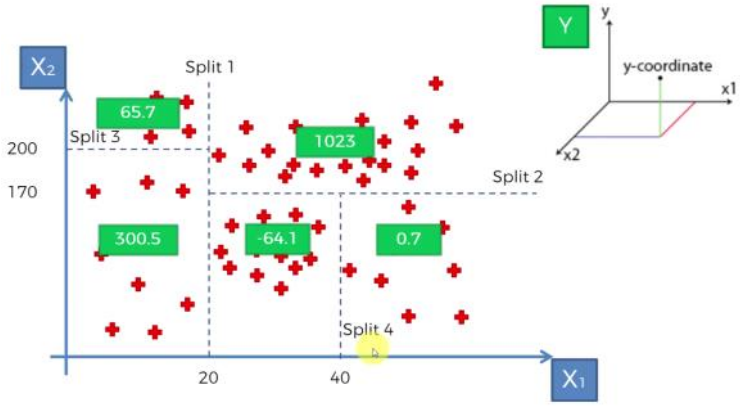
Suppose X1 and X2 are features (independent variables) and Y is the dependent variable. Y is a projection towards the screen. We are viewing the axes from top.



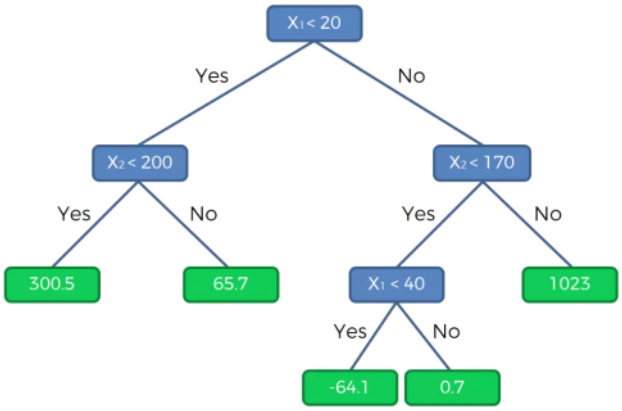
Decision tree algorithm divides the scatterplot into segments. Splits are made based on information entropy. These splits are made as long as value is added to the group of points or the points reach a 5% threshold.



The final leaves are called terminal leaves. For any new point, the algorithm makes a prediction which is equal to the average of Y of the points in the segment.



The resulting decision tree will be as follows



Python

Data pre-processing

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('Position_Salaries.csv')
X = dataset.iloc[:, 1:2].values
y = dataset.iloc[:, 2].values
```

Fitting Decision tree model to the dataset

```
# Fitting Decision Tree to the Dataset
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state=0)
regressor.fit(X, y)

# Plotting the model
plt.scatter(X, y, color = 'red')
plt.plot(X, y, color = 'green')
plt.plot(X, regressor.predict(X), color = 'blue')
plt.title('Salary Vs Level')
plt.xlabel('Level')
plt.ylabel('Salary')
plt.show()
```

Here, we have a problem. To observe this, we plot the values the model predicts over a range of intervals. The result is

```
X_grid = np.arange(min(X), max(X), 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X, y, color = 'green')
plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
plt.title('Salary Vs Level')
plt.xlabel('Level')
plt.ylabel('Salary')
plt.show()
```

This is not correct but this is what decision tree outputs owing to how it works. Note :- Decision Tree is a non-linear and non-continuous regression model. Decision tree model is useful when the number of dimensions are huge and the amount of data is greater.

For predicting a new point, we use

```
# Predicting value of a new point
prediction = regressor.predict(np.array([[6.5]]))
print(prediction)
```

Result - 150000

R

For using Regression Trees, we need to use the rpart package.
For downloading, use

```
Install.packages('rpart')
```

For using the package, use the command

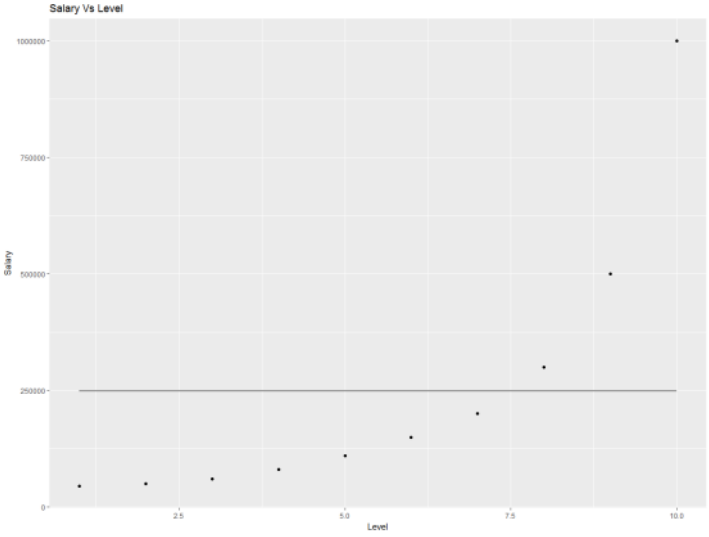
```
Library(rpart)
```

```
# Decision Tree Regression
dataset = read.csv('Position_Salaries.csv')
dataset = dataset[2:3]

library(rpart)
regressor = rpart(formula = Salary ~ ., data = dataset)

library(ggplot2)
ggplot() + geom_point(aes(dataset$Level, dataset$Salary)) +
  geom_line(aes(dataset$Level, predict(regressor, data = dataset))) +
  ggtitle('Salary Vs Level') + xlab('Level') + ylab('Salary')
```

This yields a result as below



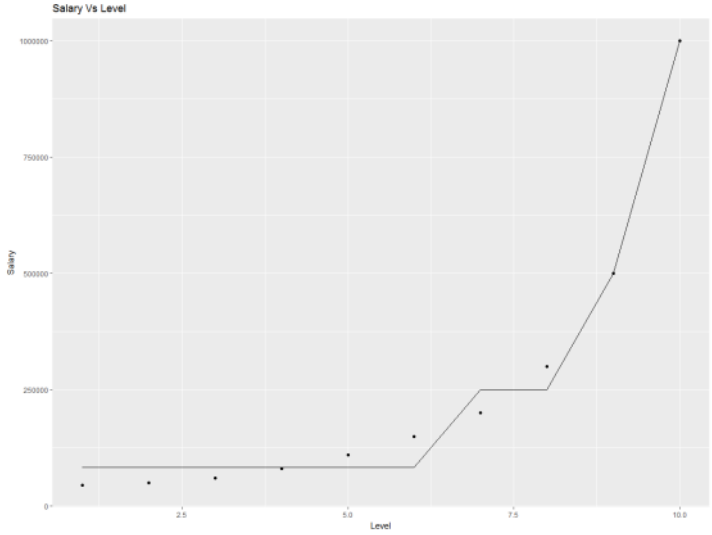
This is wrong. There is just a single regression line which predicts the same Salary for any Level.
Decision Tree does not require feature scaling owing to the way decision tree works. The problem here is owing to the number of splits. Here, there are no splits and hence we have an average for all the Levels. This is absolutely not the desired prediction model.
We hence set the control to having a minsplit of 1.

Minsplit - the minimum number of observations that must exist in a node in order for a split to be attempted.

```
# Regressor with setting control
library(rpart)
regressor = rpart(formula = Salary ~ .,
```

```
                    data = dataset,
                    control = rpart.control(minsplit = 1))
```

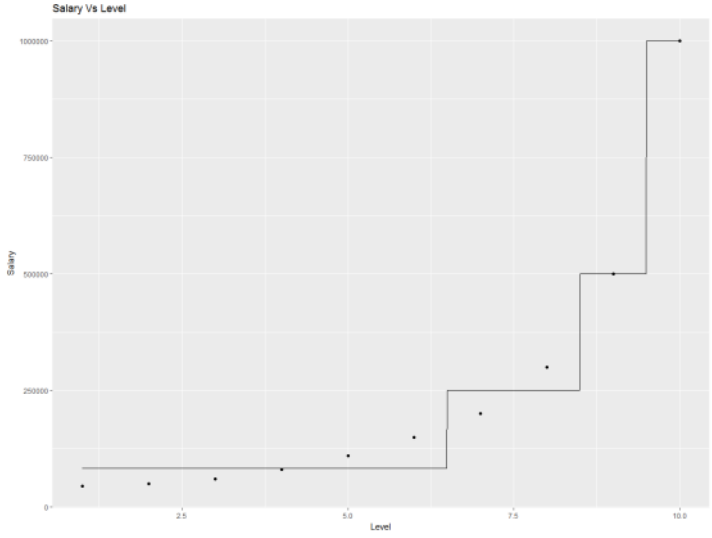
This results in the following decision tree



This may seem appealing but it's a trap. Remember Decision Tree is a non-continuous model. Hence, while plotting, we need to plot the model across a range of values and not just the values of levels in the input data to see how the model is performing for Levels not mentioned in the input data which is being used for training the model.

```
x_grid = seq(min(dataset$Level), max(dataset$Level), 0.01)
ggplot() + geom_point(aes(dataset$Level, dataset$Salary)) +
  geom_line(aes(x_grid, predict(regressor, newdata = data.frame(Level = x_grid)))) +
  ggtitle('Salary Vs Level') + xlab('Level') + ylab('Salary')
```

This provides the result



Which is the actual representation of the Decision Tree Model. Note that decision tree models for 1D, will have a sharp jump between values for various segments. We need to check the prediction across a range of values and with great resolution to get an understanding of how the model performs in that range.
Making a prediction using this, we get

```
prediction = predict(regressor, data.frame(Level = 6.5))
```

Result - 250000