**Overfitting**

Overfitting is a practical problem while building a decision tree model. The model is having an issue of overfitting is considered when the algorithm continues to go deeper and deeper in reduce the training set error but results with an increased test set error i.e., Accuracy of prediction for our model goes down. It generally happens when it builds many branches due to outliers and irregularities in data.

Two approaches which we can use to avoid overfitting are:

* Pre-Pruning
* Post-Pruning

**Pre-Pruning**

In pre-pruning, it stops the tree construction bit early. It is preferred not to split a node if its goodness measure is below a threshold value. But it’s difficult to choose an appropriate stopping point.

**Post-Pruning**

In post-pruning first, it goes deeper and deeper in the tree to build a complete tree. If the tree shows the overfitting problem then pruning is done as a post-pruning step. We use a cross-validation data to check the effect of our pruning. Using cross-validation data, it tests whether expanding a node will make an improvement or not.

If it shows an improvement, then we can continue by expanding that node. But if it shows a reduction in accuracy then it should not be expanded i.e., the node should be converted to a leaf node.

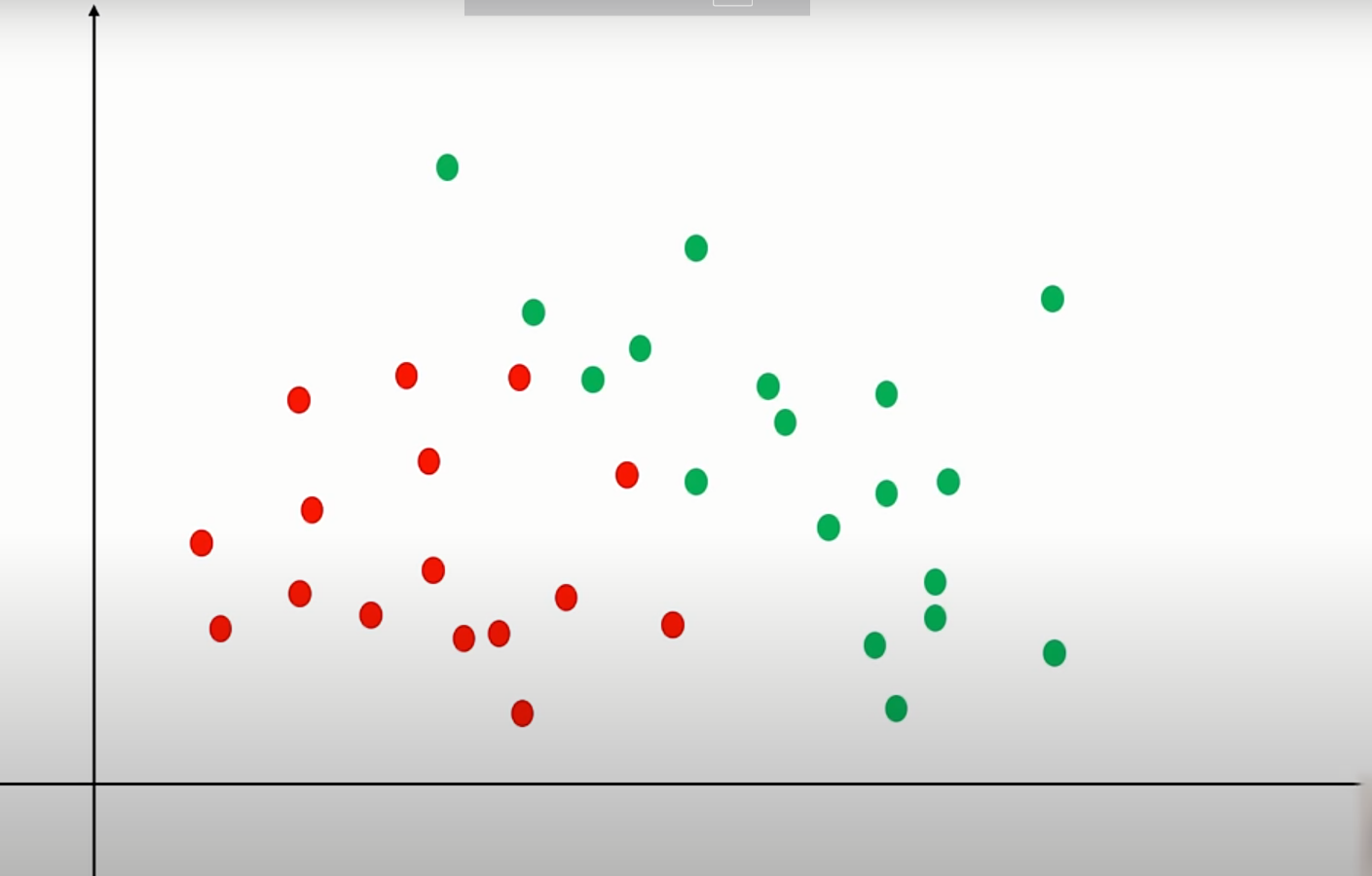
**Advantages:**

* Decision Trees `. It results in a set of rules.
* It follows the same approach as humans generally follow while making decisions.
* Interpretation of a complex Decision Tree model can be simplified by its visualizations. Even a naive person can understand logic.
* The Number of hyper-parameters to be tuned is almost null.

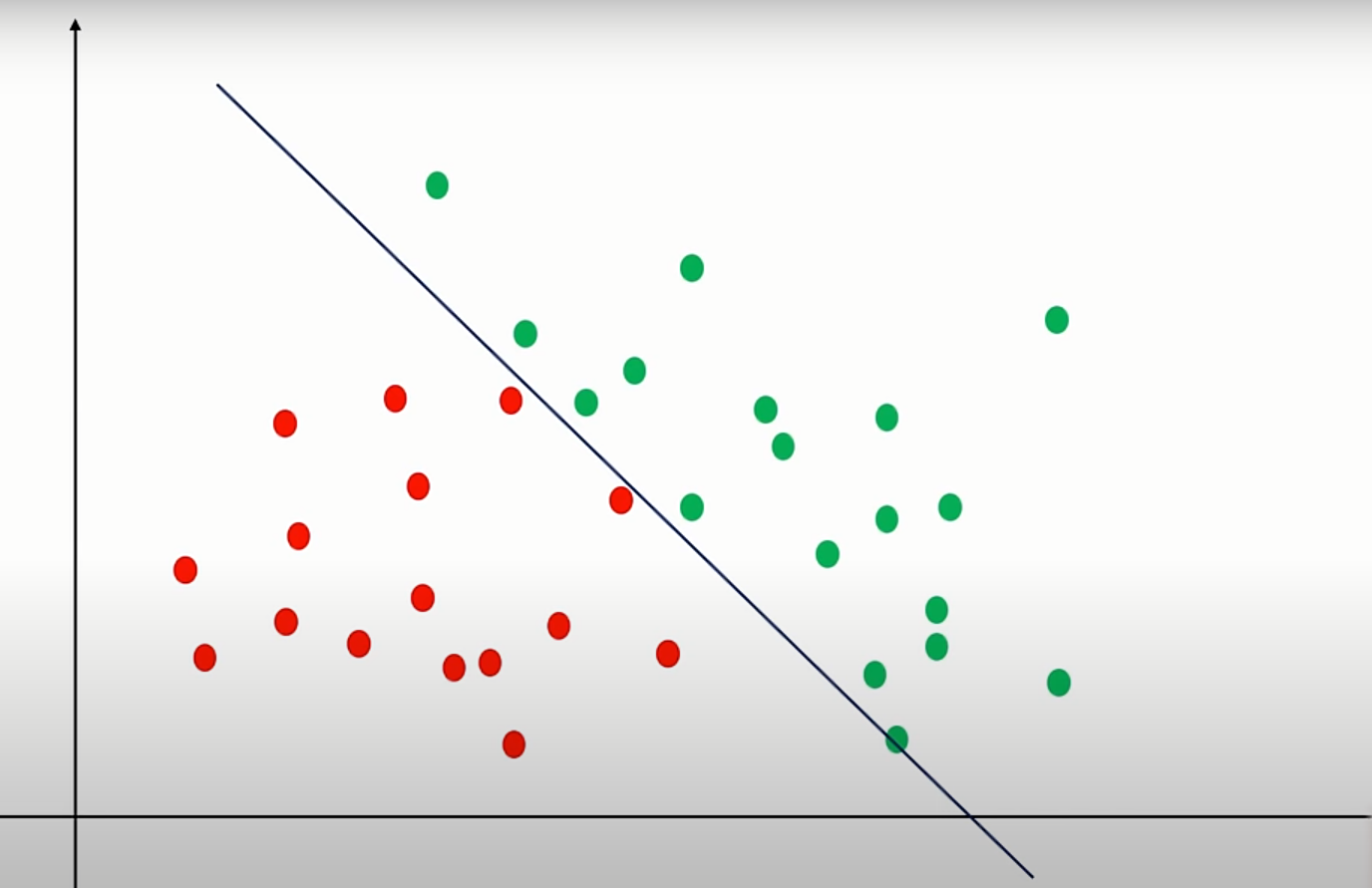
**Disadvantages:**

* There is a high probability of overfitting in Decision Tree.
* Generally, it gives low prediction accuracy for a dataset as compared to other machine learning algorithms.
* Information gain in a decision tree with categorical variables gives a biased response for attributes with greater no. of categories.
* Calculations can become complex when there are many class labels.

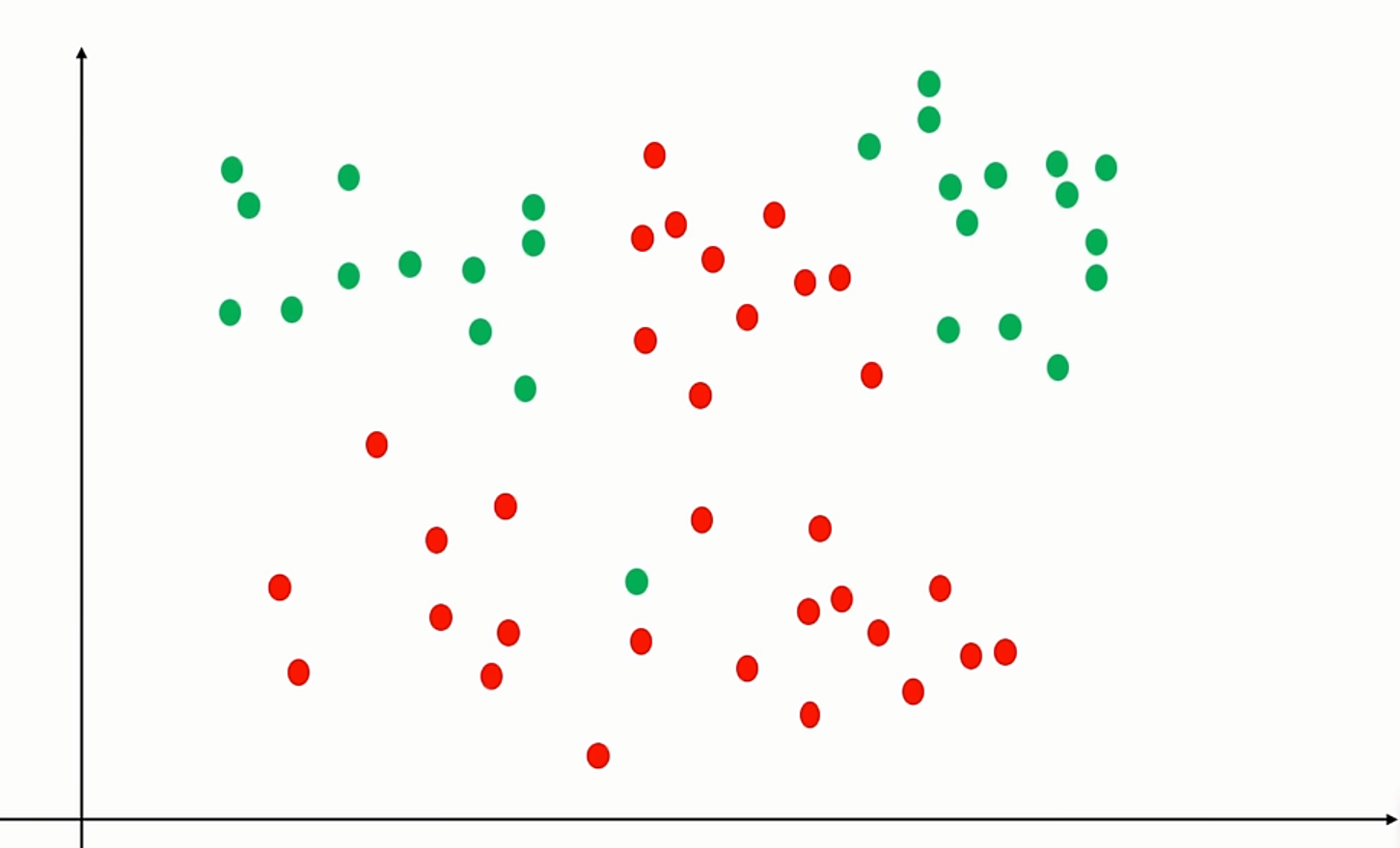
**When data set looks like**



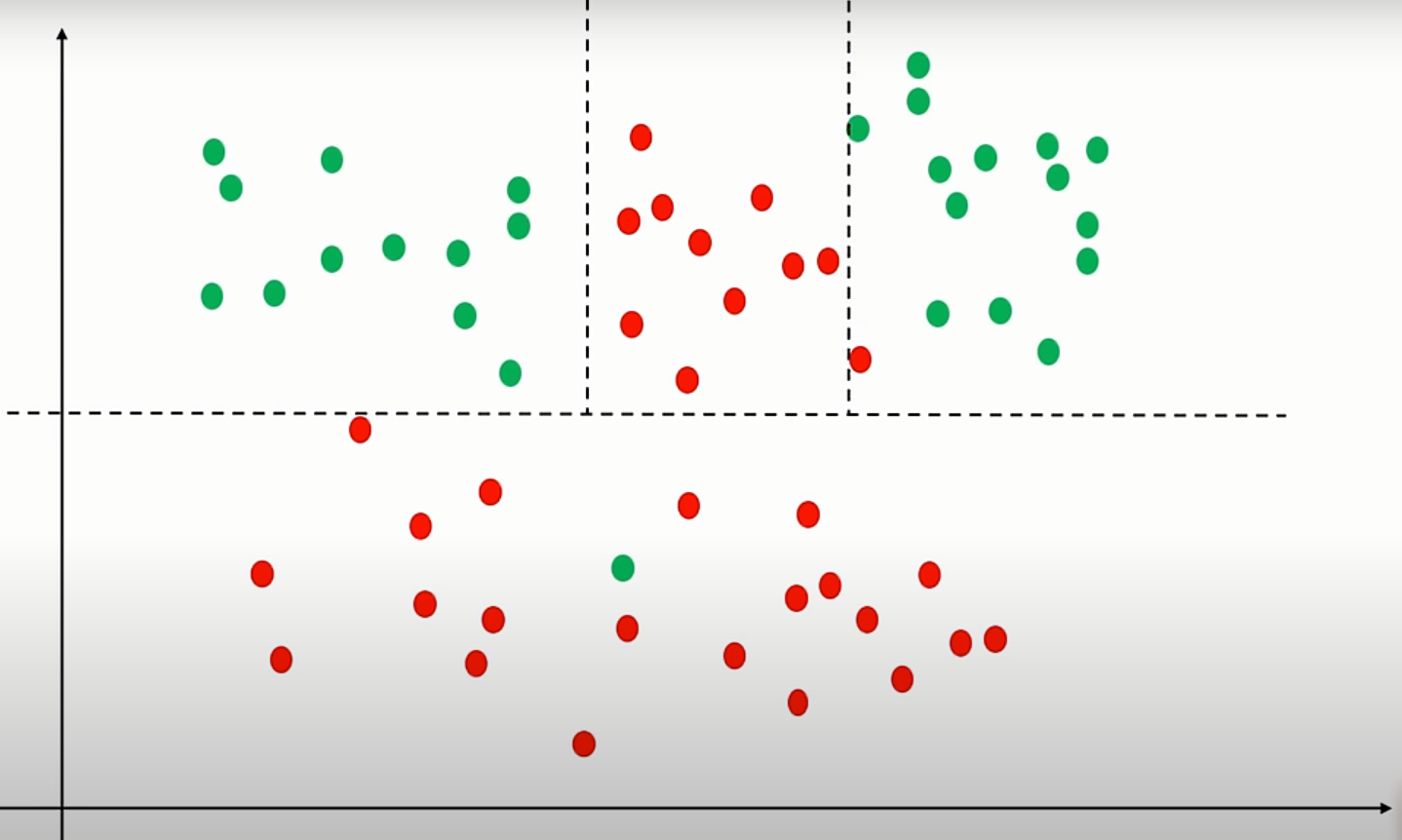
**Easily we do as follows:**



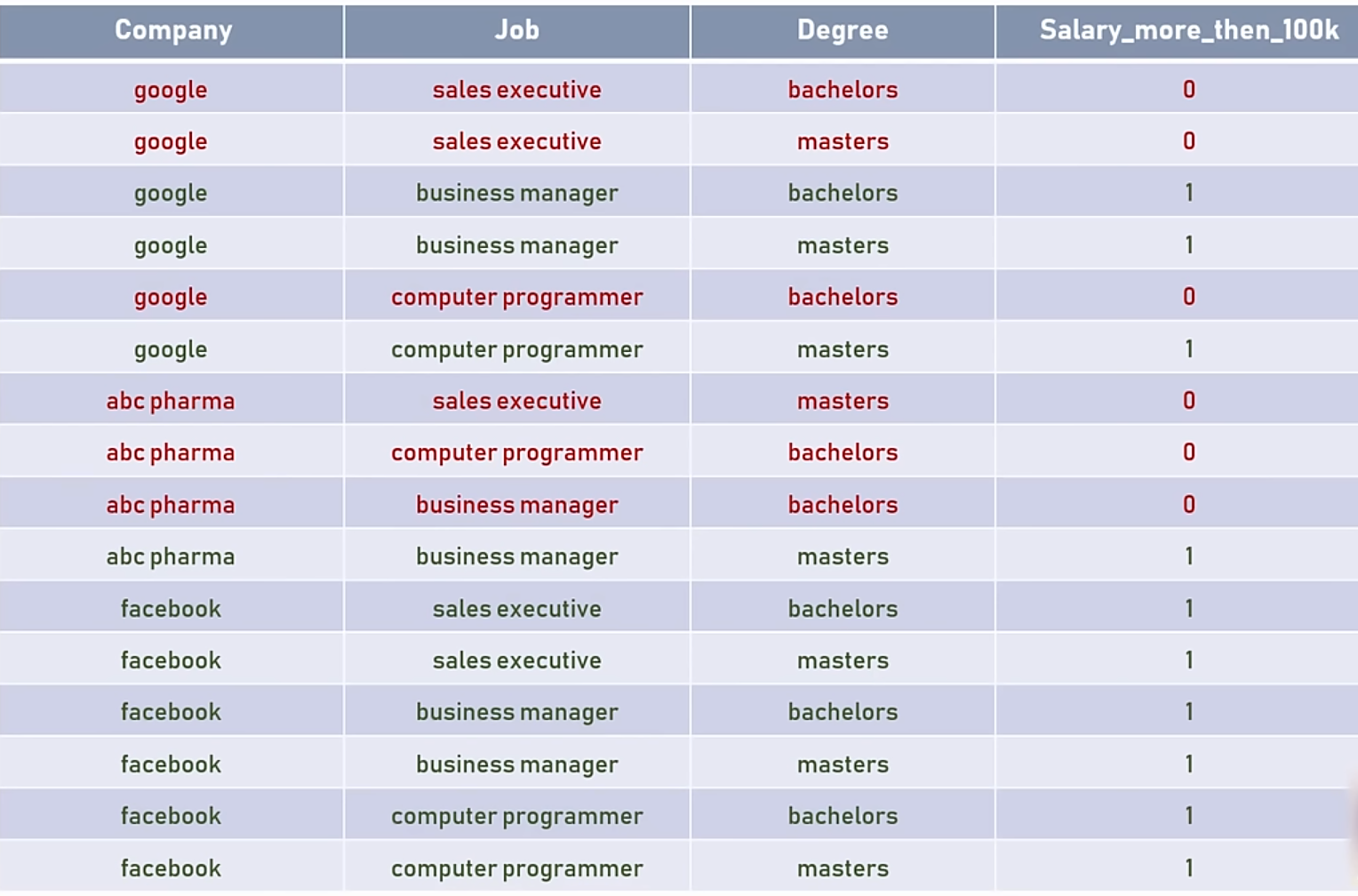
**If data set is complex as follows:**

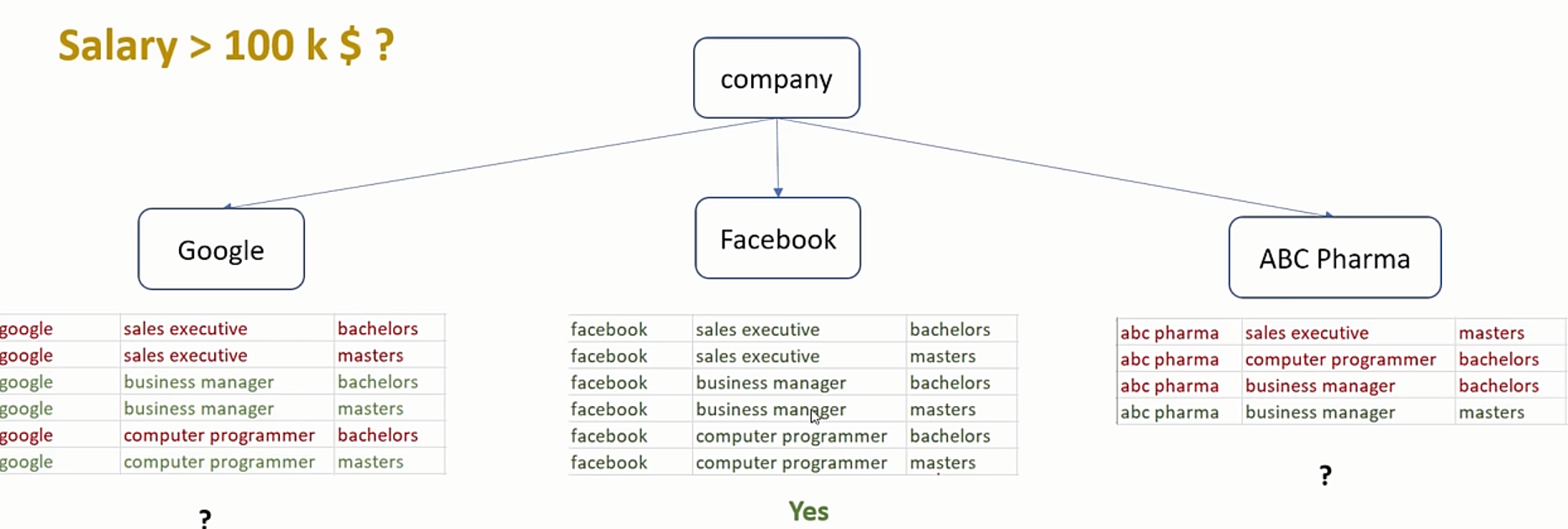
****

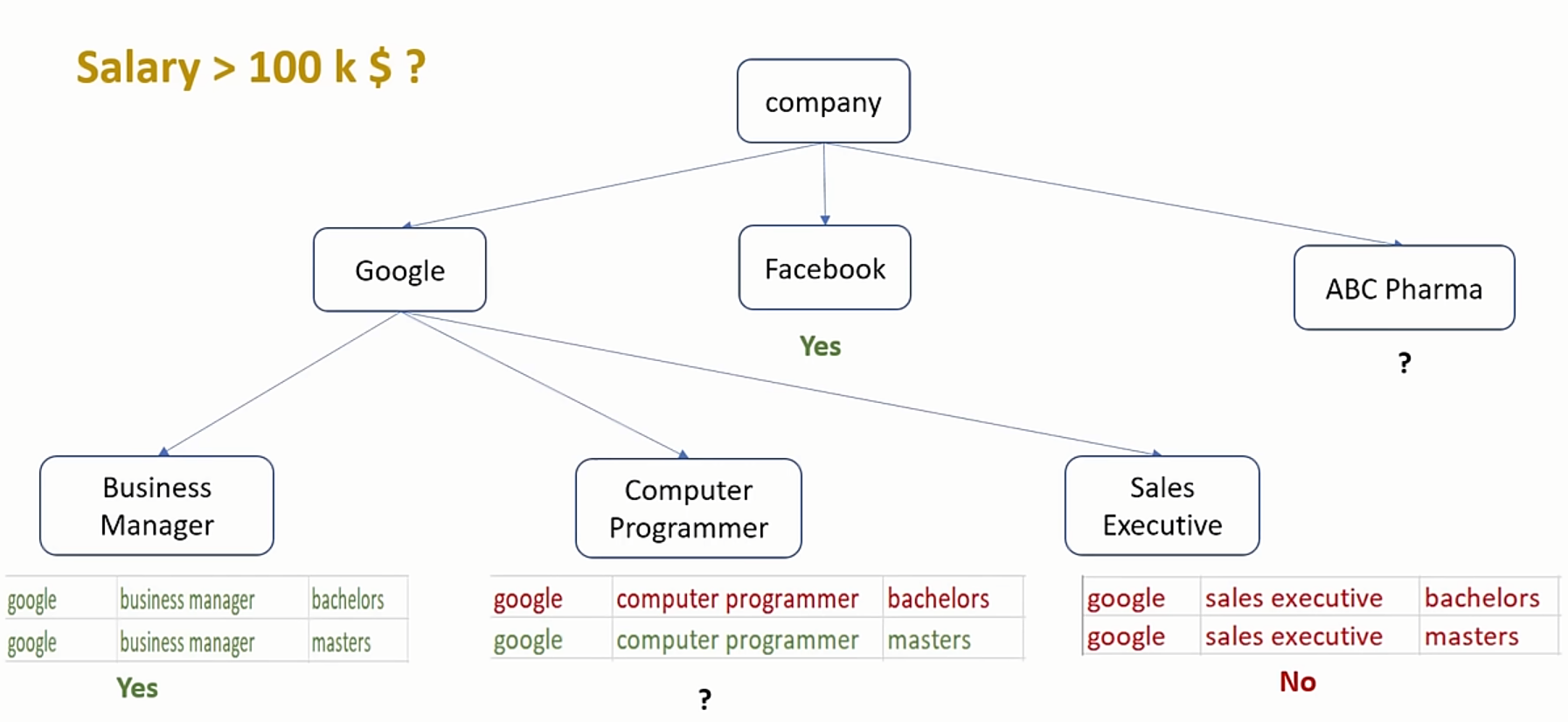
**To do as follows decision tree is req:**

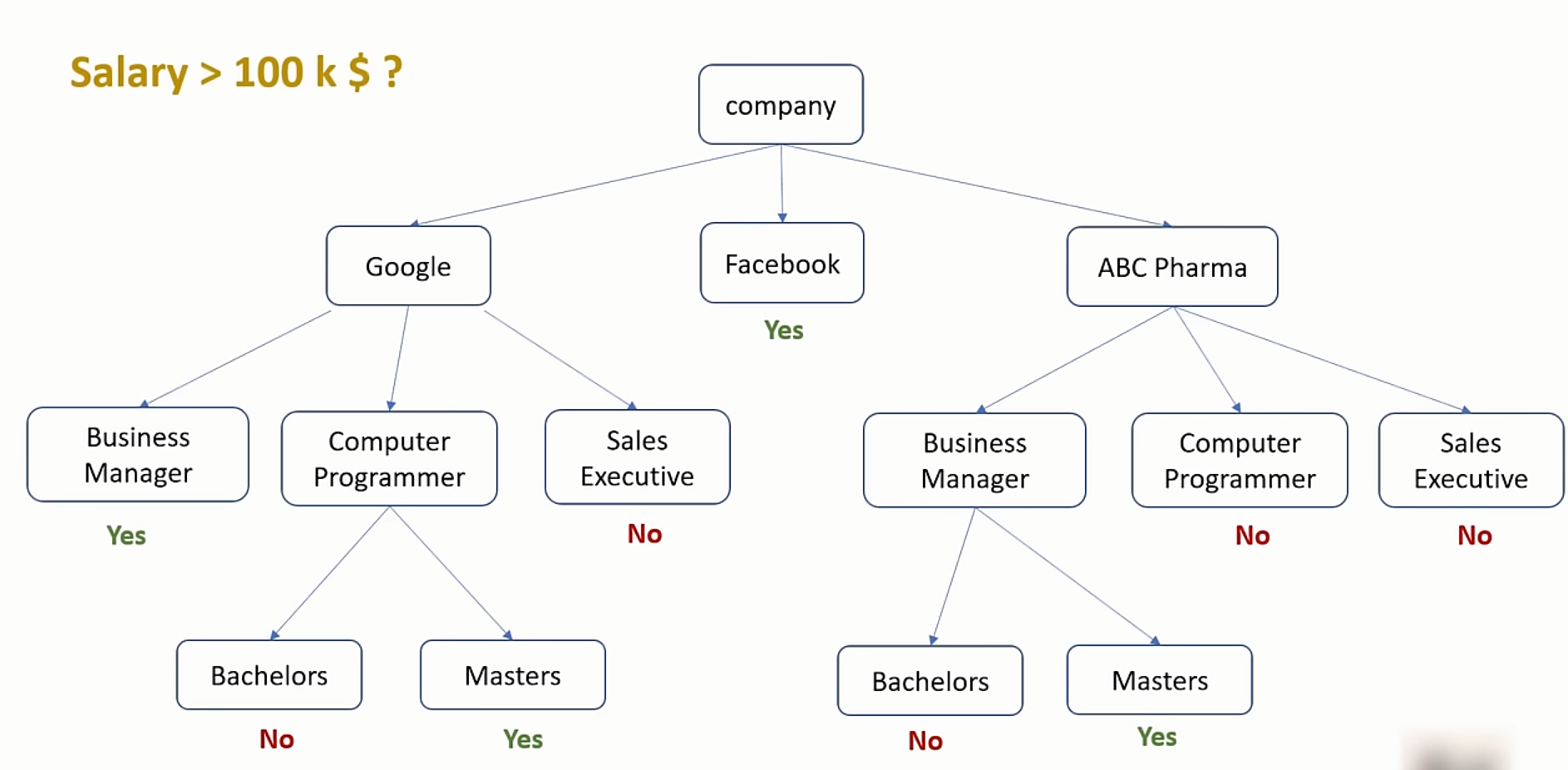
****

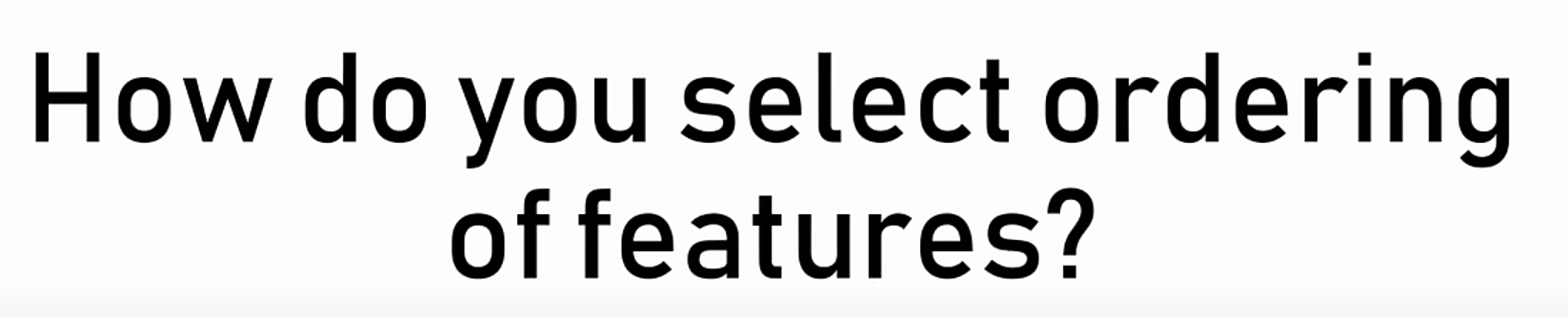
**salaries.csv**

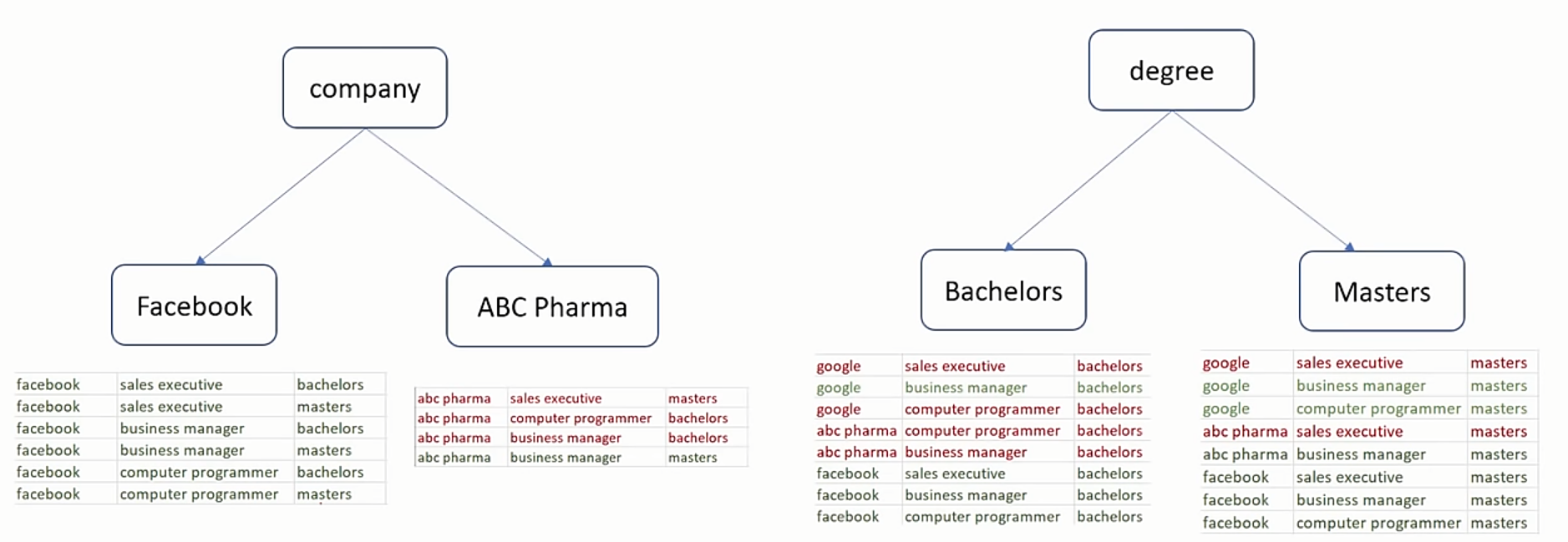
****

****

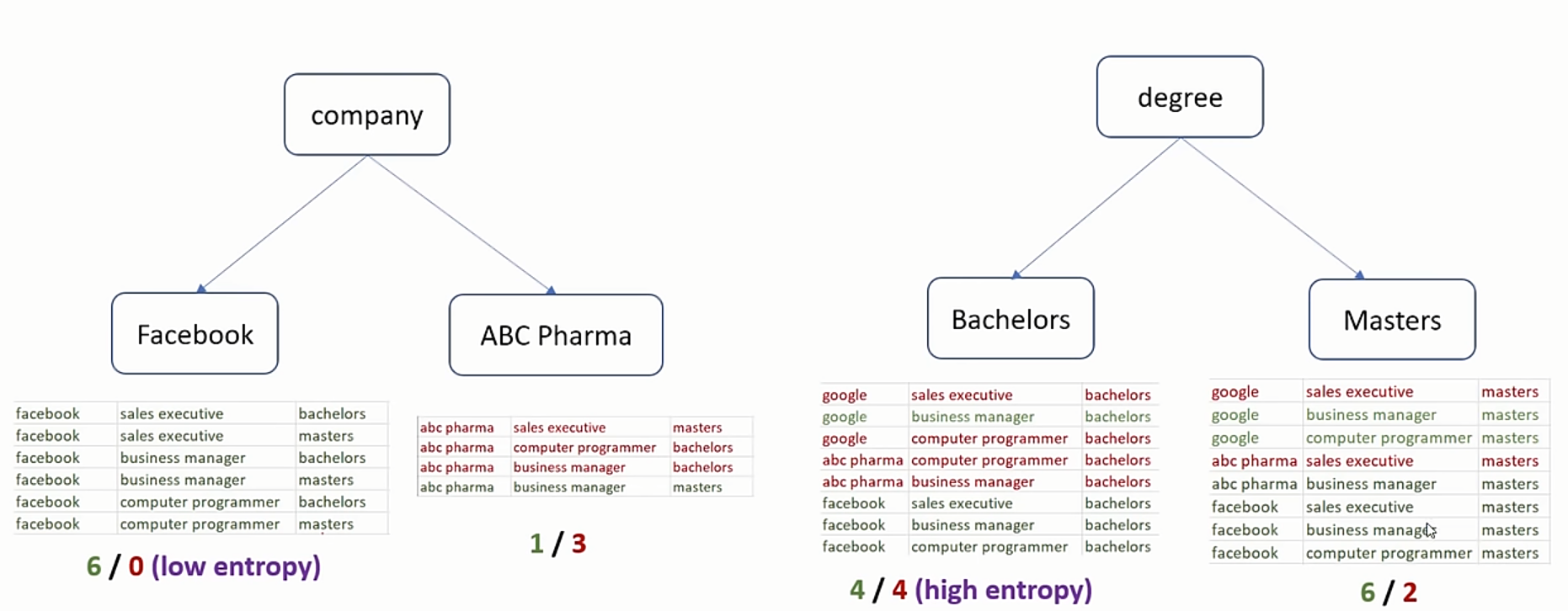
****

****

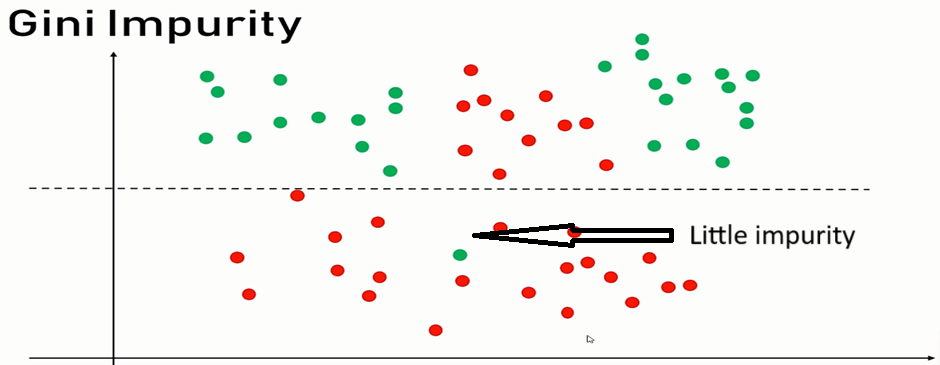
****

****

**Entropy means the measure of randomness**

****

** **

****

**PYCODE:**

#Loading Libs.

import pandas as pd

import numpy as np

# Loading Dataset

df = pd.read\_csv("salaries.csv")

#Display the data from head section

df.head(10)

#call independent variable, drop dependent (Target variable)

inputs = df.drop('salary\_more\_then\_100k',axis='columns')

target = df['salary\_more\_then\_100k']

inputs

target

#Machine can understand numbers not labels(categorical)

#Label encoding

from sklearn.preprocessing import LabelEncoder

le\_company = LabelEncoder()

le\_job = LabelEncoder()

le\_degree = LabelEncoder()

#Creating new cols

inputs['company\_n'] = le\_company.fit\_transform(inputs['company'])

inputs['job\_n'] = le\_job.fit\_transform(inputs['job'])

inputs['degree\_n'] = le\_degree.fit\_transform(inputs['degree'])

#show the cols

inputs.head(10)

#drop label cols

inputs\_n = inputs.drop(['company','job','degree'],axis='columns')

inputs\_n

#encoded as abc pharma-0, facebook-1, google-2

#encoded as Sales Executive, Business Manager, and Computer Programmer

#train the classifier

from sklearn import tree

model = tree.DecisionTreeClassifier()

model.fit(inputs\_n, target)

#show the score

model.score(inputs\_n,target)

model.predict([[2,2,1]])

model.predict([[2,0,1]])

**Exercise: Build decision tree model to predict survival based on certain parameters**



CSV file is available to download at https://github.com/rajsir/py/blob/master/ML/9\_decision\_tree/Exercise/titanic.csv

In this file using following columns build a model to predict if person would survive or not,

Pclass

Sex

Age

Fare