solving a classification problem using a decision tree algorithm

we will use this data set to predict if a person salary is more than $100000 based on the company, job title, degree and from the data, we split starting from company 1st, job title and lastly the degree.

1. how do we select the ordering of these features? if selected from the company, there is a high info gain and low entropy, while chosen from degree, there is low info gain, and high entropy
2. use an approach that gives high info at every split.

In [1]:
```
import pandas as pd
company_detail=pd.read_csv('salaries.csv')
```

In [2]:
```
company_detail.head()
```

Out[2]:

	company	job	degree	salary_more_then_100k
0	google	sales executive	bachelors	0
1	google	sales executive	masters	0
2	google	business manager	bachelors	1
3	google	business manager	masters	1
4	google	computer programmer	bachelors	0

In [3]:
```
#next is to divide it d data between the target variable and the independent vari

inputs = company_detail.drop('salary_more_then_100k', axis='columns')
target=company_detail['salary_more_then_100k']
#i will call the independent variable data frame as input
#dropping the target variable
```

In [4]:
```
inputs
```

Out[4]:

	company	job	degree
0	google	sales executive	bachelors
1	google	sales executive	masters
2	google	business manager	bachelors
3	google	business manager	masters
4	google	computer programmer	bachelors
5	google	computer programmer	masters
6	abc pharma	sales executive	masters
7	abc pharma	computer programmer	bachelors
8	abc pharma	business manager	bachelors
9	abc pharma	business manager	masters
10	facebook	sales executive	bachelors
11	facebook	sales executive	masters
12	facebook	business manager	bachelors
13	facebook	business manager	masters
14	facebook	computer programmer	bachelors
15	facebook	computer programmer	masters

In [5]:
```
#printing target
target
```

Out[5]:
```
0     0
1     0
2     1
3     1
4     0
5     1
6     0
7     0
8     0
9     1
10    1
11    1
12    1
13    1
14    1
15    1
Name: salary_more_then_100k, dtype: int64
```

#knowing that ML only understand number so we have to convert the independent variable or the rest of the columns into numbers

#2. one of the ways is to introduce label encoder.

In [6]:
```
from sklearn.preprocessing import LabelEncoder
```

In [7]:
```
#creating the object of the class, label encoder with 3 object cos there are 3 co
le_company=LabelEncoder()
le_job=LabelEncoder()
le_degree=LabelEncoder()
```

In [8]:
```
#creating one more column in the input data frame

inputs['company_n']=le_company.fit_transform(inputs['company'])
inputs['job_n']=le_company.fit_transform(inputs['job'])
inputs['degree_n']=le_company.fit_transform(inputs['degree'])

#this will create extra columns to represent your data into numbers ie label enco
inputs.head(10)
```

Out[8]:

	company	job	degree	company_n	job_n	degree_n
0	google	sales executive	bachelors	2	2	0
1	google	sales executive	masters	2	2	1
2	google	business manager	bachelors	2	0	0
3	google	business manager	masters	2	0	1
4	google	computer programmer	bachelors	2	1	0
5	google	computer programmer	masters	2	1	1
6	abc pharma	sales executive	masters	0	2	1
7	abc pharma	computer programmer	bachelors	0	1	0
8	abc pharma	business manager	bachelors	0	0	0
9	abc pharma	business manager	masters	0	0	1

#before the we used label encoder to encode the columns into numbers

#next is to drop those columns data frame

#for company: google=2, abc= 0, facebook=1

#SALES EXECUTive =2, business manager = 0, computer programmer =1

#for degree: bachelor = 0, master = 1

In [9]:
```
inputs_n=inputs.drop(['company','job','degree'],axis='columns')
```

In [10]:
```
inputs_n
```

Out[10]:

	company_n	job_n	degree_n
0	2	2	0
1	2	2	1
2	2	0	0
3	2	0	1
4	2	1	0
5	2	1	1
6	0	2	1
7	0	1	0
8	0	0	0
9	0	0	1
10	1	2	0
11	1	2	1
12	1	0	0
13	1	0	1
14	1	1	0
15	1	1	1

from the output above, it is deduced that 2 =google, 0 = abc pharmacy 1= facebook

In [11]:
```
#now we are ready to train our classifier
from sklearn import tree # cos its a decision tree
```

In [12]:
```
model=tree.DecisionTreeClassifier()
```

In [13]:
```
#train your model
model.fit(inputs_n, target)
```

Out[13]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

In [14]:
```
#predict our score by supplying the inputs_n and target data set

model.score(inputs_n, target)

# the result is 1.0 becasue the same result used for training is what was used to
```

Out[14]:
```
1.0
```

```
from the accurracy above, it can be said that my model is overfitting.
but following the fact that i am working with a small data set, it can be taken
```

In [15]:
```
#to do some prediction, let's predict person data work in google, with masters ar
model.predict([[2,2,1]]) # this is using the label encoded number to represent it
```

Out[15]:
```
array([0], dtype=int64)
```

In [16]:
```
#the value is 0, which means his salary is not more 100k
```

In [ ]:
```

```