Decision Tree Classification

We are going to solve a Classification problem using a Decision Tree algorithm

So, when you have a dataset like this, its easier to draw a decision boundry using Logistic regression

Image Alt Text

But if your dataset is complex like below then you cant just draw a single line

Image Alt Text

So you might need to split your data again and again to come up with boundaries and this is what Decision tree algo does for you
Image Alt Text

How Decision Tree works

We will use a dataset where we try to predict if a person salary is more than 100k based on the company their working, job title and degree.

So, in Decision tree, first you will split your datasets to make decisions, for example here we first split our datasets into the companies like this

Image Alt Text

So, if your company is Facebook, no matter what job title or degree you have, a person will always earn a 100k, but in the other two company, its mixed, so you need to ask further questions, basically splitting it more, for example, splitting it now based on Job title
Image Alt Text

As we can see above, if a person who works in Google is a Business Manager then no matter what degree they will always earn 100k, and if the person is Sales executive then they will never earn 100k but for a Computer programmer its mixed again so we can again split it into more.
So basically, this is how Decision tree work, you keep splitting untill you come up with something like this

Image Alt Text

No this may sound simple as we only have three attributes, but in real world you might have like 50, so it matters in what order you split them, above example we first split them with

Company then job title and then degree.

In which order you select to split your attr will impact the performance of your model

Order of splitting

Now, how do we exactly select the ordering of these features?

Lets look at the example, on left, we split Company first and on the oher we split degree first

Image Alt Text

Now, if we observe, on the Company one, we are getting a little bit of a pure subset on 'Facebook', meaning all the sample are green, i.e the same,
So this have a very low entrophy, entrophy basically means the measure of randomness in your sample, so in 'Facebook' there is no randomness since everything is green, meaning it has low entrophy which is good!

Then on 'ABC Pharma' there is some entrophy but majority of them are red.

While as you can see on the Degree one, both the sub catogory has mixed green and red meaning it has high entrophy which is not good, so overall, Splitting Company first would be better.

So due to this, on the left hand side, we will have a High information gain and on right side a Low information gain, hence, u should do the one first which has High info gain

Image Alt Text

Gini Impurity

There is another term that you hear often when dealing with Decision tree which is Gini Impurity, this is nothing but an impurities in your dataset, for example below, if we split our dataset, you can see in the below part theres a single green dot, meaning it is a little bit impure but its fine since most of them are red, so it is almost pure but there's a lil bit of impurity

Image Alt Text

Its sort of similar to entrophy but you can find the differences yourself by looking it up

Coding Part

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

In [4]:
```
data = pd.read_csv('salaries.csv')
data
```

Out[4]:

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

In [7]:
```
X = data.drop(['salary_more_then_100k'], axis=1)
y = data['salary_more_then_100k']
```

Now, we need to convert company,job and degree columns to numbers since its text.

One of the ways is to use Label Encoder

In [8]:
```
from sklearn.preprocessing import LabelEncoder
```

Since we have 3 of these columns to Encode, we need to create 3 objects for each of them

In [9]:
```
le_company = LabelEncoder()
le_job = LabelEncoder()
le_degree = LabelEncoder()
```

Now, just call the fit and tranform method on the Company, job and title column and add a new column for each encoded

In [10]:
```
X['company_n'] = le_company.fit_transform(X['company'])
X['job_n'] = le_company.fit_transform(X['job'])
X['degree_n'] = le_company.fit_transform(X['degree'])
```

In [12]:
```
X.head()
```

Out[12]:

	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

Now since we got the label encoded, we dont need those columns anymore, so just create a new DF and drop the company, job and degree columns

In [16]:
```
new_X = X.drop(['company', 'job', 'degree'], axis=1)
new_X
```

Out[16]:

	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

In [17]:
```
from sklearn import tree
```

```
In [22]: model = tree.DecisionTreeClassifier()
```

```
In [23]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(new_X, y, test_size=0.2)
```

```
In [24]: print(len(X_train))
         print(len(X_test))

12
4
```

```
In [26]: model.fit(X_train, y_train)

Out[26]: ▼ DecisionTreeClassifier

         DecisionTreeClassifier()
```

```
In [27]: model.score(new_X, y)

Out[27]: 0.9375
```

```
In [28]: model.predict([[2, 0, 0]])

C:\Users\User\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklea
rn\base.py:465: UserWarning: X does not have valid feature names, but Decisio
nTreeClassifier was fitted with feature names
  warnings.warn(

Out[28]: array([1], dtype=int64)
```

```
In [32]: predict = model.predict(X_test)
         predict

Out[32]: array([1, 1, 1, 1], dtype=int64)
```

Creating a new Dataframe storing actual and predicted value as column

```
In [34]: obj = {
             "company_n" : X_test['company_n'],
             "job_n" : X_test['job_n'],
             "degree_n" : X_test['degree_n'],
             "Actual Value" : y_test,
             "Predicted Value" : predict
         }

         newdf = pd.DataFrame(obj)
         newdf
```

Loading [MathJax]/extensions/Safe.js

Out[34]:

	company_n	job_n	degree_n	Actual Value	Predicted Value
5	2	1	1	1	1
13	1	0	1	1	1
4	2	1	0	0	1
3	2	0	1	1	1