In [1]:

```python
import pandas as pd
```

In [2]:

```python
df = pd.read_csv("datatestonehotencoding.csv")
```

In [4]:

```python
print 'done step 1'
```

```
done step 1
```

In [5]:

```python
inputs = df.drop(['Result','Difference'] , axis="columns")
results = df['Result']
```

In [9]:

```python
def unique_vals(rows, col):
    return set(row[col] for row in rows)
```

In [11]:

```python
unique_vals(inputs.values,0)
```

Out[11]:

```
{0, 1}
```

In [12]:

```python
unique_vals(results.values,0)
```

Out[12]:

```
{'D', 'L', 'W'}
```

In [13]:

```python
dfarr = df.values
```

In [19]:

```
def class_counts(rows):
    counts = {}
    for row in rows:
        label = row[8]
        if label not in counts:
            counts[label] = 0
        counts[label] += 1
    return counts
```

In [20]:

```
class_counts(dfarr)
```

Out[20]:

```
{'D': 4, 'L': 7, 'W': 13}
```

In [21]:

```
def is_numeric(value):
    return isinstance(value,int) or isinstance(value,float)
```

In [26]:

```
class Question:
    def __init__(self,column,value):
        self.column = column
        self.value = value
    def match(self,a_sample):
        #the val is only numeric here
        val = a_sample[self.column]
        return val >= self.value
```

In [27]:

```
q = Question(0,0.5)
```

In [28]:

```
q.match(dfarr[0])
```

Out[28]:

```
True
```

In [29]:

```
q.match(dfarr[1])
```

Out[29]:

```
False
```

In [31]:

```
def partition(rows,question):
    true_rows,false_rows = [],[]
    for row in rows :
        if question.match(row):
            true_rows.append(row)
        else:
            false_rows.append(row)
    return true_rows , false_rows
```

In [32]:

```
true_rows , false_rows = partition(dfarr , Question(0,0.5))
```

```
In [33]:
```

```
true_rows
```

```
Out[33]:
```

```
[array([1, 0, 1, 0, 1, 0, 0, 0, 'D', 0], dtype=objec
t),
 array([1, 0, 0, 0, 1, 1, 0, 0, 'W', 2], dtype=objec
t),
 array([1, 0, 0, 0, 1, 1, 0, 0, 'W', 2], dtype=objec
t),
 array([1, 0, 1, 1, 0, 0, 0, 0, 'W', 1], dtype=objec
t),
 array([1, 1, 1, 0, 0, 0, 0, 0, 'L', -2], dtype=obje
ct),
 array([1, 0, 1, 0, 0, 0, 0, 1, 'D', 0], dtype=objec
t),
 array([1, 0, 0, 1, 0, 0, 0, 1, 'W', 1], dtype=objec
t),
 array([1, 1, 1, 0, 0, 0, 0, 0, 'L', -2], dtype=obje
ct),
 array([1, 0, 1, 0, 0, 0, 0, 1, 'D', 0], dtype=objec
t),
 array([1, 1, 0, 1, 0, 0, 0, 0, 'W', 2], dtype=objec
t),
 array([1, 0, 1, 0, 0, 0, 0, 1, 'W', 1], dtype=objec
t),
 array([1, 0, 0, 1, 0, 0, 0, 1, 'W', 1], dtype=objec
t),
 array([1, 1, 1, 0, 0, 0, 0, 0, 'L', -1], dtype=obje
ct),
 array([1, 1, 1, 0, 0, 0, 0, 0, 'L', -2], dtype=obje
ct)]
```

```
In [34]:
```

```
false_rows
```

```
Out[34]:
```

```
[array([0, 1, 1, 0, 0, 1, 0, 0, 'L', -2], dtype=obje
ct),
 array([0, 0, 1, 0, 1, 1, 0, 0, 'W', 3], dtype=objec
t),
 array([0, 0, 1, 0, 1, 1, 0, 0, 'W', 3], dtype=objec
t),
 array([0, 1, 0, 0, 1, 1, 0, 0, 'W', 2], dtype=objec
t),
 array([0, 1, 1, 0, 1, 0, 0, 0, 'W', 1], dtype=objec
t),
 array([0, 1, 1, 0, 0, 1, 0, 0, 'L', -2], dtype=obje
ct),
 array([0, 1, 1, 0, 0, 1, 0, 0, 'W', 1], dtype=objec
t),
 array([0, 1, 1, 0, 0, 0, 1, 0, 'D', 0], dtype=objec
t),
 array([0, 1, 1, 0, 0, 1, 0, 0, 'L', -3], dtype=obje
ct),
 array([0, 1, 0, 1, 0, 0, 0, 1, 'W', 3], dtype=objec
t)]
```

```
In [37]:
```

```
true_rows , false_rows = partition(dfarr , Question(4,0.5))
```

In [38]:

```
true_rows
```

Out[38]:

```
[array([1, 0, 1, 0, 1, 0, 0, 0, 'D', 0], dtype=objec
t),
 array([0, 0, 1, 0, 1, 1, 0, 0, 'W', 3], dtype=objec
t),
 array([1, 0, 0, 0, 1, 1, 0, 0, 'W', 2], dtype=objec
t),
 array([0, 0, 1, 0, 1, 1, 0, 0, 'W', 3], dtype=objec
t),
 array([1, 0, 0, 0, 1, 1, 0, 0, 'W', 2], dtype=objec
t),
 array([0, 1, 0, 0, 1, 1, 0, 0, 'W', 2], dtype=objec
t),
 array([0, 1, 1, 0, 1, 0, 0, 0, 'W', 1], dtype=objec
t)]
```

```
In [39]:
```

```
false_rows
```

```
Out[39]:
```

```
[array([0, 1, 1, 0, 0, 1, 0, 0, 'L', -2], dtype=obje
ct),
 array([0, 1, 1, 0, 0, 1, 0, 0, 'L', -2], dtype=obje
ct),
 array([0, 1, 1, 0, 0, 1, 0, 0, 'W', 1], dtype=objec
t),
 array([0, 1, 1, 0, 0, 0, 1, 0, 'D', 0], dtype=objec
t),
 array([0, 1, 1, 0, 0, 1, 0, 0, 'L', -3], dtype=obje
ct),
 array([1, 0, 1, 1, 0, 0, 0, 0, 'W', 1], dtype=objec
t),
 array([1, 1, 1, 0, 0, 0, 0, 0, 'L', -2], dtype=obje
ct),
 array([1, 0, 1, 0, 0, 0, 0, 1, 'D', 0], dtype=objec
t),
 array([1, 0, 0, 1, 0, 0, 0, 1, 'W', 1], dtype=objec
t),
 array([1, 1, 1, 0, 0, 0, 0, 0, 'L', -2], dtype=obje
ct),
 array([1, 0, 1, 0, 0, 0, 0, 1, 'D', 0], dtype=objec
t),
 array([1, 1, 0, 1, 0, 0, 0, 0, 'W', 2], dtype=objec
t),
 array([1, 0, 1, 0, 0, 0, 0, 1, 'W', 1], dtype=objec
t),
 array([0, 1, 0, 1, 0, 0, 0, 1, 'W', 3], dtype=objec
t),
 array([1, 0, 0, 1, 0, 0, 0, 1, 'W', 1], dtype=objec
t),
 array([1, 1, 1, 0, 0, 0, 0, 0, 'L', -1], dtype=obje
ct),
 array([1, 1, 1, 0, 0, 0, 0, 0, 'L', -2], dtype=obje
ct)]
```

In [42]:

```
def gini(rows):
    countsdictionary = class_counts(rows)
    impurity = 1
    for key in countsdictionary:
        prob_of_thisresult = countsdictionary[key]/float(len(rows
        impurity -= prob_of_thisresult**2
    return impurity
```

In [43]:

```
gini(dfarr)
```

Out[43]:

```
0.59375
```

In [44]:

```
def info_gain(left,right,current_uncertainty):
    p = float(len(left)/ (len(left) + len(right)))
    return current_uncertainty - p*gini(left) - (1-p)*gini(right)
```

In [46]:

```
no_of_players = len(dfarr[0]) - 2
no_of_players
```

Out[46]:

```
8
```

In [81]:

```
def find_best_split(rows):
    best_gain = 0
    best_player_to_split = -1
    best_question = None
    current_uncertainty = gini(rows)
    noplayers = no_of_players

    for col in range(noplayers):
        question = Question(col,0.5)
        tr,fr = partition(rows,question)
        if len(tr) == 0 or len(fr) == 0:
            continue
        gain = info_gain(tr,fr,current_uncertainty)
        if(gain > best_gain):
            best_gain,best_question,best_player_to_split = gain,q
    return best_gain,best_question
    #could return best player_id also here
```

In [82]:

```
bf,bq = find_best_split(dfarr)
```

In [83]:

```
#here it is showing Xhaka because we passed the entire input rath
bp
```

Out[83]:

```
2
```

In [59]:

```
df.columns
```

Out[59]:

```
Index([u'Guendouzi', u'Torreira', u'Xhaka', u'Willoc
k', u'Ramsey', u'Ozil',
       u'Mkhitaryan', u'Ceballos', u'Result', u'Diff
erence'],
      dtype='object')
```

In [60]:

```
df.columns[1]
```

Out[60]:

```
'Torreira'
```

In [61]:

```
df.columns[2]
```

Out[61]:

```
'Xhaka'
```

In [67]:

```
class Leaf:
    def __init__(self,rows):
        self.playerid = 'Leaf Node'
        self.predictions = class_counts(rows)
```

In [68]:

```
class Decision_Node:
    def __init__(self,question,true_branch,false_branch):
        self.playerid = question.column
        self.question = question
        self.true_branch = true_branch
        self.false_branch = false_branch
```

In [69]:

```
def build_tree(rows):
    gain,question = find_best_split(rows)
    if gain == 0:
        return Leaf(rows)
    trs,frs = partition(rows,question)
    true_branch = build_tree(trs)
    false_branch = build_tree(frs)
    return Decision_Node(question,true_branch,false_branch)
```

In [86]:

```
def traverse(decnodeorleaf):
    if(not isinstance(decnodeorleaf,Decision_Node)):
        return
    else:
        print decnodeorleaf.playerid
        traverse(decnodeorleaf.true_branch)
        traverse(decnodeorleaf.false_branch)
```

In [87]:

```
traverse(5)
```

In [88]:

```
rootnode = build_tree(dfarr)
```

In [89]:

```
traverse(rootnode)
```

```
2
1
6
4
5
0
3
7
```

In [ ]:

```

```

In [ ]:

```

```