Bayes Theorem - Tennis Data set

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import networkx as nx
```

Load Data Set

In [2]:
```
data = pd.read_csv('./dataset/tennis.csv')
data
```

Out[2]:

	outlook	temp	humidity	windy	play
0	sunny	hot	high	False	no
1	sunny	hot	high	True	no
2	overcast	hot	high	False	yes
3	rainy	mild	high	False	yes
4	rainy	cool	normal	False	yes
5	rainy	cool	normal	True	no
6	overcast	cool	normal	True	yes
7	sunny	mild	high	False	no
8	sunny	cool	normal	False	yes
9	rainy	mild	normal	False	yes
10	sunny	mild	normal	True	yes
11	overcast	mild	high	True	yes
12	overcast	hot	normal	False	yes
13	rainy	mild	high	True	no

Outlook vs Play

In [3]:
```
group = data[['outlook','play']].groupby(by=['outlook'])
```

In [4]:
```
group.count()
```

Out[4]:

	play
outlook	
overcast	4
rainy	5
sunny	5

In [5]:
```
table = pd.crosstab(data['play'],columns=data['outlook'])
table
```

Out[5]:

outlook	overcast	rainy	sunny
play			
no	0	2	3
yes	4	3	2

In [6]:
```
table.loc['total'] = table.loc['no'] + table.loc['yes']
table['total'] = table['overcast'] + table['rainy']  + table['sunny']
```

Pivot Table

In [7]:
```
table
```

Out[7]:

outlook	overcast	rainy	sunny	total
play				
no	0	2	3	5
yes	4	3	2	9
total	4	5	5	14

Contingency Table

In [8]:
```
prob = table.div(14)# table.div(len(data)), table.div(table.iloc[-1,-1])
prob
```

Out[8]:

outlook	overcast	rainy	sunny	total
play				
no	0.000000	0.142857	0.214286	0.357143
yes	0.285714	0.214286	0.142857	0.642857
total	0.285714	0.357143	0.357143	1.000000

Marginal Probability:

- $P(overcast)$
- $P(rainy)$
- $P(sunny)$

In [9]:
```
# P(overcast)
p_o = prob['overcast'].total
# P(rainy)
p_r = prob['rainy'].total
# P(sunny)
p_s = prob['sunny'].total
```

In [10]:
```
# print
print('Probability of overcast: P(overcast) = %0.3f'%p_o )
print('Probability of rainy: P(rainy) = %0.3f'%p_r )
print('Probability of sunny: P(sunny) = %0.3f'%p_s )
```

```
Probability of overcast: P(overcast) = 0.286
Probability of rainy: P(rainy) = 0.357
Probability of sunny: P(sunny) = 0.357
```

Marginal Probability:

- $P(no)$
- $P(yes)$

In [11]:
```
# P(no)
p_no = prob.loc['no'].total
# P(yes)
p_yes = prob.loc['yes'].total
```

In [12]:
```
# print
print('Probability of No: P(no) = %0.3f'%p_no )
print('Probability of Yes: P(yes) = %0.3f'%p_yes )
```

```
Probability of No: P(no) = 0.357
Probability of Yes: P(yes) = 0.643
```

Conditionality Probability

$$P(A|B) = \frac{P(AandB)}{P(B)}$$

- $P(Sunny|Yes)$
- $P(Sunny|No)$
- $P(Sunny|Yes)$
- $P(Sunny|No)$

In [13]:
```
prob
```

Out[13]:

outlook	overcast	rainy	sunny	total
play				
no	0.000000	0.142857	0.214286	0.357143
yes	0.285714	0.214286	0.142857	0.642857
total	0.285714	0.357143	0.357143	1.000000

In [14]:
```
prob.keys()
```

Out[14]:
```
Index(['overcast', 'rainy', 'sunny', 'total'], dtype='object', name='outlook')
```

In [15]:
```
prob.index
```

Out[15]:
```
Index(['no', 'yes', 'total'], dtype='object', name='play')
```

In [16]:
```
prob['overcast']['yes'] # joint probability
```

Out[16]:
```
0.2857142857142857
```

In [17]:
```
prob['total']['yes'] # marginal probability
```

Out[17]:
```
0.6428571428571429
```

In [18]:
```
prob['overcast']['total'] # marginal probability
```

Out[18]:
```
0.2857142857142857
```

In [19]:
```
def jointprob(A,B,table):
    """
    jointprob(A,B) will return probability of combination attribute from
    contigency table. P(A and B)
    A = column
    B = row
    >>> jointprob(A,B,table)

    """
    return table[A][B]#.loc[B]

def marginprob(B,table):
    """
    marginprob(B) will return probability of attribute from
    contigency table. P(B)
    B = row
    >>> marginprob(B,table)

    """
    return table.loc[B][-1]
```

In [20]:
```
# P(overcast and no)
jointprob('rainy','no',prob)
```

Out[20]:
```
0.14285714285714285
```

In [21]:
```
p_sunny_given_yes = jointprob('sunny','yes',prob) / p_yes
print('Probability of sunny given yes: P(sunny|yes) = %0.3f'%p_sunny_given_yes)
p_sunny_given_no = jointprob('sunny','no',prob) / p_no
print('Probability of sunny given no: P(sunny|no) = %0.3f'%p_sunny_given_no)
print('\n')
p_overcast_given_yes = jointprob('overcast','yes',prob) / p_yes
print('Probability of overcast given yes: P(overcast|yes) = %0.3f'%p_overcast_giv
p_overcast_given_no = jointprob('overcast','no',prob) / p_no
print('Probability of overcast given no: P(overcast|no) = %0.3f'%p_overcast_given
print('\n')
p_rainy_given_yes = jointprob('rainy','yes',prob) / p_yes
print('Probability of rainy given yes: P(overcast|yes) = %0.3f'%p_rainy_given_yes
p_rainy_given_no = jointprob('rainy','no',prob) / p_no
print('Probability of rainy given no: P(overcast|no) = %0.3f'%p_rainy_given_no)
```

```
Probability of sunny given yes: P(sunny|yes) = 0.222
Probability of sunny given no: P(sunny|no) = 0.600


Probability of overcast given yes: P(overcast|yes) = 0.444
Probability of overcast given no: P(overcast|no) = 0.000


Probability of rainy given yes: P(overcast|yes) = 0.333
Probability of rainy given no: P(overcast|no) = 0.400
```

Probability Tree

In [22]:
```
table = prob
```

In [23]:
```
start = table.index.name
ind1 = 'P(%s)=%0.3f'%(table.index[0],marginprob(table.index[0],table))
ind2 = 'P(%s)=%0.3f'%(table.index[1],marginprob(table.index[1],table))
# Given index-1 probability of events
event11 = '%s=%0.3f'%(prob.keys()[0],jointprob(prob.keys()[0],prob.index[0],prob)
event12 = '%s=%0.3f'%(prob.keys()[1],jointprob(prob.keys()[1],prob.index[0],prob)
event13 = '%s=%0.3f'%(prob.keys()[2],jointprob(prob.keys()[2],prob.index[0],prob)
# Given index-2 probability of events
event21 = '%s=%0.3f'%(prob.keys()[0],jointprob(prob.keys()[0],prob.index[1],prob)
event22 = '%s=%0.3f'%(prob.keys()[1],jointprob(prob.keys()[1],prob.index[1],prob)
event23 = '%s=%0.3f'%(prob.keys()[2],jointprob(prob.keys()[2],prob.index[1],prob)
```

In [24]:
```
drawData = {'from':[start,ind1,ind1,ind1,start,ind2,ind2,ind2],
            'to':[ind1,event11,event12,event13,ind2,event21,event22,event23]}
draw = pd.DataFrame(drawData)
draw
```

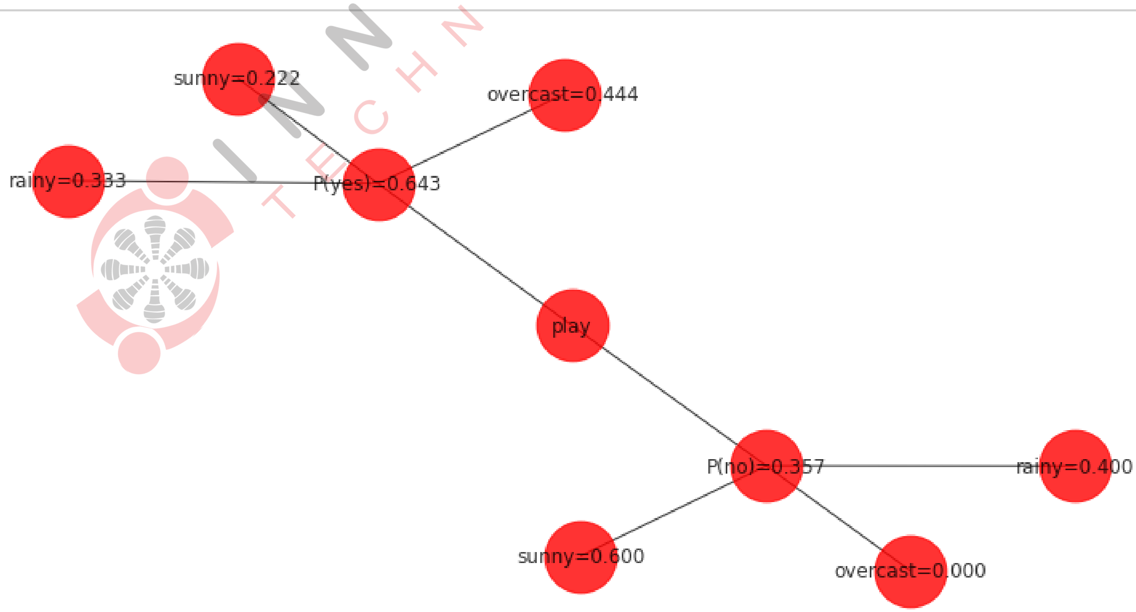
Out[24]:

	from	to
0	play	P(no)=0.357
1	P(no)=0.357	overcast=0.000
2	P(no)=0.357	rainy=0.400
3	P(no)=0.357	sunny=0.600
4	play	P(yes)=0.643
5	P(yes)=0.643	overcast=0.444
6	P(yes)=0.643	rainy=0.333
7	P(yes)=0.643	sunny=0.222

In [25]:
```
fig = plt.figure(figsize=(10,5))

# Build your graph. Note that we use the DiGraph function to create the graph!
G=nx.from_pandas_edgelist(draw, 'from', 'to')

# Make the graph
nx.draw(G, with_labels=True, node_size=2000,alpha=0.8, arrows=True,linewitdh=50.0
```



Classification Report

In [26]:
```
prob
```

Out[26]:

outlook	overcast	rainy	sunny	total
play				
no	0.000000	0.142857	0.214286	0.357143
yes	0.285714	0.214286	0.142857	0.642857
total	0.285714	0.357143	0.357143	1.000000

Bayes Theorem

$$P(A|B) = \frac{P(A) * A(B|A)}{P(B)}$$

Example:

$$P(yes|rainy) = \frac{P(yes) * A(rainy|yes)}{P(rainy)}$$

$$= \frac{P(yes) * A(rainy|yes)}{P(yes) * P(rainy|yes) + p(no) * P(rainy|no)}$$

In [41]:
```
def conditional(column,row,table):
    """
    conditional(column,row,table)
    """
    return jointprob(column,row,table)/marginprob(row,table)
```

In [56]:
```
Event = 'yes'
given = 'rainy'

p_event_given = marginprob('yes',prob) * conditional('rainy','yes',prob) / margin
```

In [57]:
```
p_event_given
```

Out[57]: 0.3333333333333333