Title: 12.2 Exercises

Author: Chad Wood

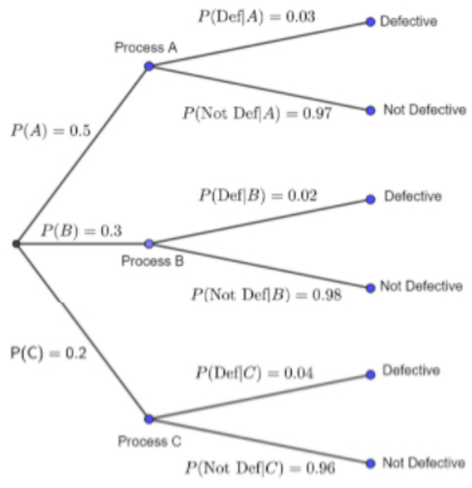
Date: 27 Feb 2022

Modified By: Chad Wood

Description: This program demonstrates the use of python to calculate probabilities and entropy of data.

Calculating Probabilities using Bayes' Theorem

A computer graphics card is manufactured using three different processes, A, B, and C. The following tree diagram gives the probability that a graphics card is manufactured with each process and the probability of defective and non-defective cards from each process.



Use Bayes' Theorem to calculate the following probabilities. These can be done by hand or in Python.

(a) If a randomly chosen graphics card is defective, what is the probability it was manufactured using Process A?
(b) If a randomly chosen graphics card is not defective, what is the probability it was manufactured using Process C?

In [45]:
```python
import pandas as pd

builds = ['A', 'B', 'C']
a = [0.5, 0.03, 0.97]
b = [0.3, 0.02, 0.98]
c = [0.2, 0.04, 0.96]

prob_df = pd.DataFrame([a, b, c], builds, columns=['Probability', 'Defective', 'Not_Defective'])
```

In [70]:
```python
# Takes A, B, A(Given)B; Returns probability A|B
def bayes(A, B, B_A):
    return (B_A*A)/B


# (a)
#_____

# Likelihood built = A
A = prob_df[prob_df.index=='A'].Probability
# Likelihood of being defective
B = sum(prob_df['Defective']*prob_df['Probability'])
# Probability of being defective given A
B_A = prob_df[prob_df.index=='A'].Defective

print(f'Probability build is "A" given card is defective: {bayes(A, B, B_A)}')

# (b)
#_____

# Likelihood built = C
A = prob_df[prob_df.index=='C'].Probability
# Likelihood of not being defective
B = sum(prob_df['Not_Defective']*prob_df['Probability'])
# Probability of not being defective given C
B_A = prob_df[prob_df.index=='C'].Not_Defective

print(f'Probability build is "C" given card is not defective: {bayes(A, B, B_A)}')
```

```
Probability build is "A" given card is defective: A    0.517241
dtype: float64
Probability build is "C" given card is not defective: C    0.197734
dtype: float64
```

Entropy Function for a Probability Distribution

Create a Python function that takes in an array of probabilities p1, p2, . . . , pn, from a discrete probability distribution with n possible outcomes and returns the entropy of the corresponding random variable. Use the formula on p. 143 of Essential Math for Data Science to calculate the entropy

In [74]:
```python
import numpy as np

# Takes array of probabilities; Outputs entropy
def entropy(x):
    return - np.sum(x * np.log2(x))
```

Calculating Entropies of Probability Distributions

Two random variables X and Y can take on the values 1, 2, . . . , 5 with the following probabilities.

x	P(X = x)
1	0.2
2	0.2
3	0.2
4	0.2
5	0.2

y	P(Y = y)
1	0.1
2	0.4
3	0.1
4	0.3
5	0.1

(a) Use the function you created in the previous problem to calculate the entropies of X and Y .
(b) Compare the two values found in part (a). Which one is bigger? Explain intuitively why this is the case.

In [83]:
```python
X = np.array([0.2, 0.2, 0.2, 0.2, 0.2])
Y = np.array([0.1, 0.4, 0.1, 0.3, 0.1])

# (a)
print(f'The entropy of (X, Y) is respectively: {entropy(X), entropy(Y)}')

# (b)
print('The entropy of array Y is smaller, intuitively, because the data contains more "rare" instances than observed with X. '
      'Entropy takes advantage of being able to encode unique occurences within data while foregoing predictable data.')
```

```
The entropy of (X, Y) is respectively: (2.321928094887362, 2.046439344671015)
The entropy of array Y is smaller, intuitively, because the data contains more "rare" instances than observed with X. Entropy t
akes advantage of being able to encode unique occurences within data while foregoing predictable data.
```