Jin Young Jeon

Lab 2 Probability and Bayes Theorem

From **lab2-probability.py**
For **Task 1** see probability.py

Task 2: 6x6 Contingency Table & see code for returning possible number of instances:

yx	1	2	3	4	5	6	total
1	3	4	0	1	1	1	10
2	1	1	0	0	0	0	2
3	0	0	1	0	0	2	3
4	0	0	0	1	0	0	1
5	0	0	0	0	2	0	2
6	0	0	1	1	0	0	2
total	4	5	2	3	3	3	20

Task 3:

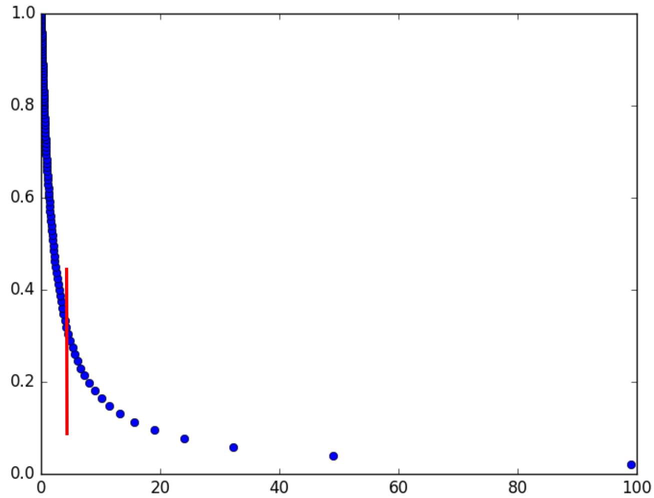
a) p(X=1|Y=2) [.5pt]
1/2 = 0.5

b) p(X=5,Y=5) [.5pt]
p(x=5)p(y=5|x=5) = 3/20 * 2/3 = 0.1

Task 4: Compute the expected value for dice 1 [.5pt] ev = 3.6
Task 5: Variance: 61.75
Task 6 Hypothetical Contingency Table:
If X and Y were independent, P(x,y)=P(x)P(y). By product rule, P(x,y)=P(x)P(y|x)=P(y)P(x|y). Now we can infer that P(x)=P(x|y) and P(y)=P(y|x). For the hypothetical table created below, P(x,y)=P(x)P(y) is satisfied.

yx	1	2	3	4	5	6	total
1	1	1	1	1	1	1	6
2	1	1	1	1	1	1	6
3	1	1	1	1	1	1	6
4	1	1	1	1	1	1	6
5	1	1	1	1	1	1	6
6	1	1	1	1	1	1	6
total	6	6	6	6	6	6	36

From **lab2-bayes.py**



Initially, neglecting the base rate, people confirm the person is more likely to be a librarian.
However, the posterior probability, the belief that the person is a librarian given the description, decreases the more data there is. It would only seem rational to choose librarian over farmer around approximately where x < 5, in which posterior probability is above 0.5.

(Appendix)

From lab2-probability.py

```
def marginal_probability(x, n): #computes marginal probability of n in list x
    k, total = 0, 0
    while k < len(x):
        if n == x[k]:
            k += 1
            total += 1
        else:
            k += 1
    return total/len(x)

def c(i, j, x, y): #returns total number of pairs that corresponds to C(i,j)
    k, total = 0, 0
    while k < len(x):
        if i == x[k]:
            if j == y[k]:
                total += 1
            k += 1
        else:
            k += 1
    return total
c(1, 3, x, y) #0
c(1, 2, x, y) #1
c(5, 5, x, y) #2

def variance(x):
    sum, var = 0, 0
    for j in range(len(x)):
        sum += x[j]
    mean = sum/(float(len(x)))
    print(mean)
    for i in range(len(x)):
        var += (x[i] - mean)**2
    return var
print (variance(x)) #61.75
```

From lab2-bayes.py

```
des1lib = 0.8
des1farm = 0.4
prior_lib = np.linspace(0.01,1,100) #prior_lib = p(lib) and p(farm) = 1 - p(lib)
prior_ratio = (1-prior_lib)/(prior_lib)
#given Bayes Theorem:
posterior_prob = ((des1lib)*(prior_lib))/((des1lib)*(prior_lib)+(des1farm)*(1-prior_lib))
plt.plot(prior_ratio, posterior_prob, 'o')
plt.show()
```