In [1]:
```
# Imports
# Portions of this code borrowed from Parth
import numpy as np
import matplotlib.pyplot as plt
import math
import csv
```

In [2]:
```
# Constants
edibility =    {'p':-1,'e':1}
cap_shape =    {'b':1,'c':2,'x':3,'f':4,'k':5,'s':6}
cap_surface =  {'f':1,'g':2,'y':3,'s':4}
cap_color =    {'n':1,'b':2,'c':3,'g':4,'r':5,'p':6,'u':7,'e':8,'w':9,'y':10}
bruises =      {'t':1,'f':2}
odor =         {'a':1,'l':2,'c':3,'y':4,'f':5,'m':6,'n':7,'p':8,'s':9}
gill_attach =  {'a':1,'d':2,'f':3,'n':4}
gill_spacing = {'c':1,'w':2,'d':3}
gill_size =    {'b':1,'n':2}
gill_color =   {'k':1,'n':2,'b':3,'h':4,'g':5,'r':6,'o':7,'p':8,'u':9,'e':10,'
w':11,'y':12}
stalk_shape =  {'e':1,'t':2}
stalk_root =   {'b':1,'c':2,'u':3,'e':4,'z':5,'r':6,'?':7}
stalk_sar =    {'f':1,'y':2,'k':3,'s':4}
stalk_sbr =    {'f':1,'y':2,'k':3,'s':4}
stalk_car =    {'n':1,'b':2,'c':3,'g':4,'o':5,'p':6,'e':7,'w':8,'y':9}
stalk_cbr =    {'n':1,'b':2,'c':3,'g':4,'o':5,'p':6,'e':7,'w':8,'y':9}
veil_type =    {'p':1,'u':2}
veil_color =   {'n':1,'o':2,'w':3,'y':4}
ring_number =  {'n':1,'o':2,'t':3}
ring_type =    {'c':1,'e':2,'f':3,'l':4,'n':5,'p':6,'s':7,'z':8}
spore_color =  {'k':1,'n':2,'b':3,'h':4,'r':5,'o':6,'u':7,'w':8,'y':9}
population =   {'a':1,'c':2,'n':3,'s':4,'v':5,'y':6}
habitat =      {'g':1,'l':2,'m':3,'p':4,'u':5,'w':6,'d':7}

params = [cap_shape, cap_surface, cap_color, bruises, odor, gill_attach, gill_
spacing, gill_size,
          gill_color, stalk_shape, stalk_root, stalk_sar, stalk_sbr, stalk_car
, stalk_cbr,
          veil_type, veil_color, ring_number, ring_type, spore_color, populati
on, habitat]
```

In [3]:
```
# Data Import/Transform
yvals = []
xvals = []

with open('./agaricus-lepiota.data', 'rt') as csvfile:
    mushrooms = csv.reader(csvfile, delimiter=',')
    for row in mushrooms:
        yvals.append(row[0])
        xvals.append(row[1:])
rows = len(yvals)
features = len(xvals[0])
print("Rows: " + str(rows))
print("Vals: " + str(features))

#transform data
poisonous = 0
edible = 0
for i in range(0, rows):
    yvals[i] = edibility[yvals[i]]
    if(yvals[i]==1): edible+=1
    else: poisonous+=1
    for j in range(0, features):
        xvals[i][j] = params[j][xvals[i][j]]
print("E: " + str(edible))
print("P: " + str(poisonous))
print(yvals[0], xvals[0])
```

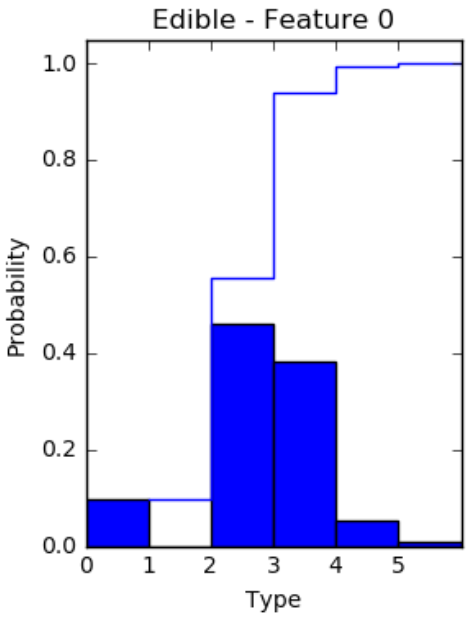
```
Rows: 8124
Vals: 22
E: 4208
P: 3916
(-1, [3, 4, 1, 1, 8, 3, 1, 2, 1, 1, 4, 4, 4, 8, 8, 1, 3, 2, 6, 1, 4, 5])
```

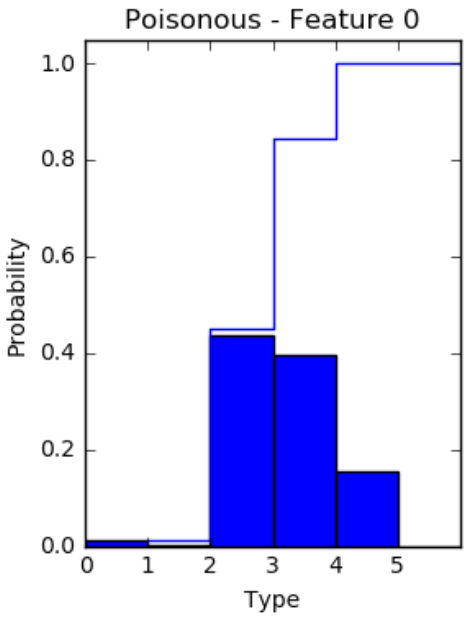
In [4]:
```
def randomize_data(X): #part a
    #np.random.seed(0)
    np.random.shuffle(X)
    num_train = int(math.floor(0.90 * X.shape[0]))
    num_test = X.shape[0] - num_train
    X_train = X[:num_train, :]
    X_test = X[-num_test:, :]
    return X_train, X_test

#changing data format
Data = np.zeros((rows, 23))
for i in range(0,rows):
    Data[i,0] = yvals[i]
    Data[i,1:]= xvals[i][:]

#getting random subset of data
X, X_test = randomize_data(Data)
train_size = X.shape[0]
print train_size

e = float((X[:,0]== 1).sum())
p = float((X[:,0]==-1).sum())
```

```
7311
```

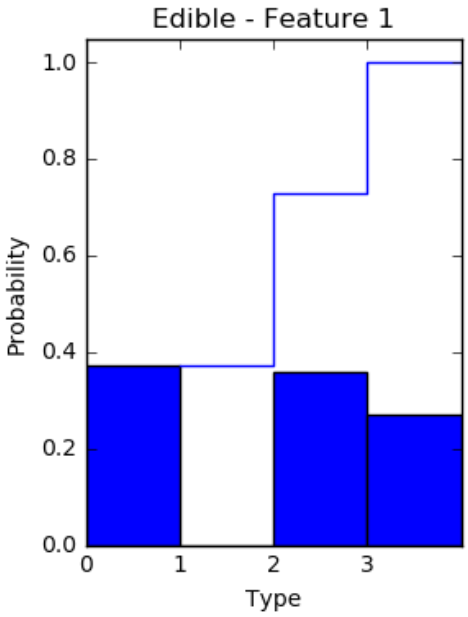
In [5]:
```
selected_features = range(0,5)
for i in selected_features:
    types = len(params[i])
    e_feat = np.zeros(types)
    p_feat = np.zeros(types)

    for typ in range(0,types):
        e_feat[typ] = ((X[:,0]== 1) & (X[:,i+1]==typ+1)).sum()/e
        p_feat[typ] = ((X[:,0]==-1) & (X[:,i+1]==typ+1)).sum()/p

    print e_feat
    print p_feat

    plt.subplot('121')
    plt.bar(range(types), e_feat, width=1)
    plt.step(range(1, types+1), np.cumsum(e_feat))
    plt.title('Edible - Feature {}'.format(i))
    plt.xlabel('Type')
    plt.ylabel('Probability')
    plt.ylim(0.0, 1.05)
    plt.xticks(range(types))

    plt.subplot('122')
    plt.bar(range(types), p_feat, width=1)
    plt.step(range(1, types+1), np.cumsum(p_feat))
    plt.title('Poisonous - Feature {}'.format(i))
    plt.xlabel('Type')
    plt.ylabel('Probability')
    plt.ylim(0.0, 1.05)
    plt.xticks(range(types))

    plt.tight_layout()
    plt.show()
```

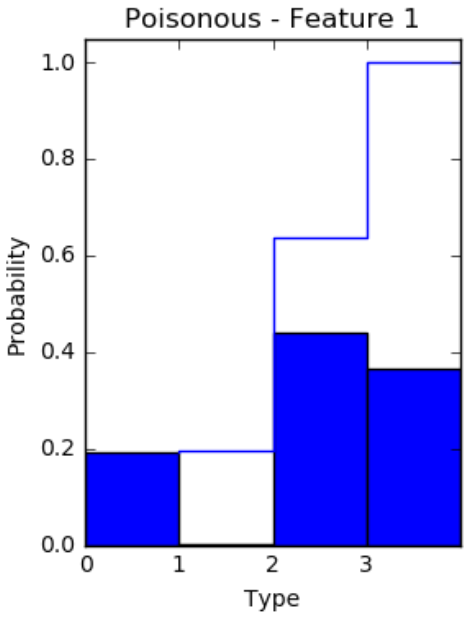
```
[ 0.09571655  0.          0.46139609  0.38180857  0.05393971  0.00713908]
[ 0.01218475  0.00113347  0.43581751  0.39501275  0.15585152  0.        ]
```



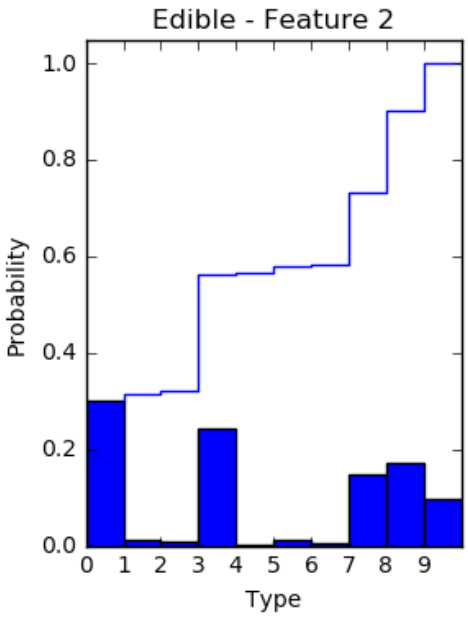


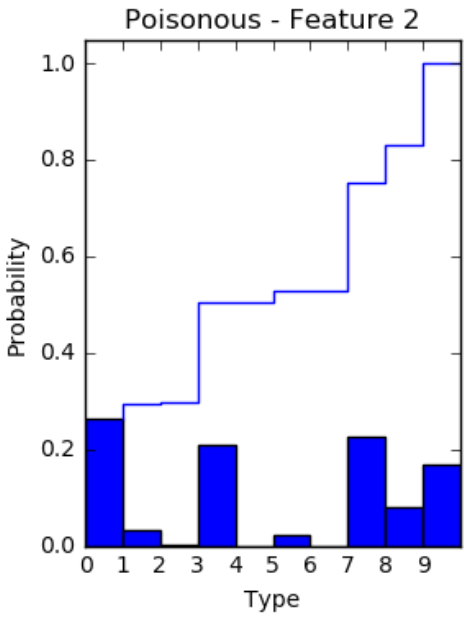

```
[ 0.37149656  0.          0.35695399  0.27154944]
[ 0.19353925  0.00113347  0.44120147  0.36412581]
```



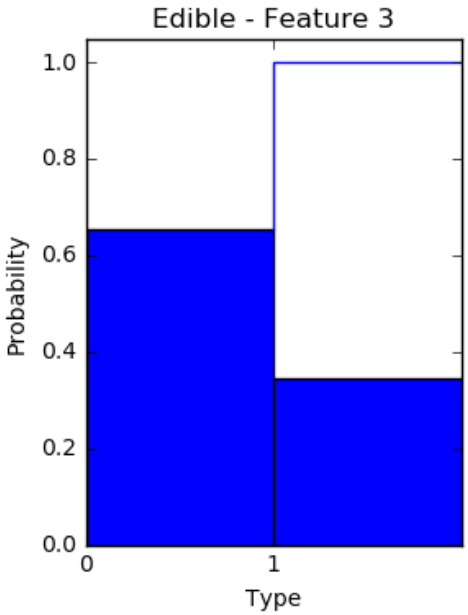


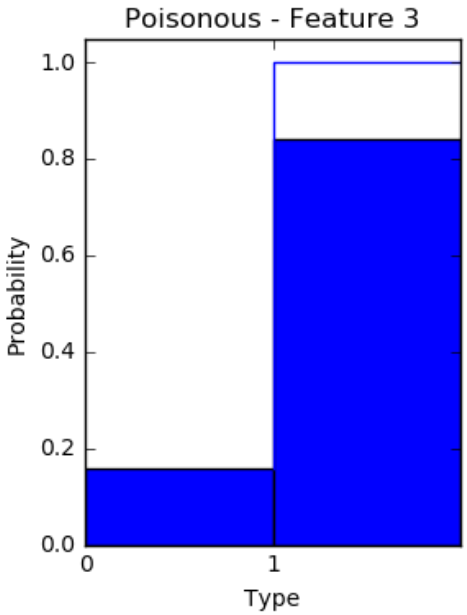

```
[ 0.30169223  0.01163406  0.00793231  0.24299313  0.00290851  0.01242729
  0.00423057  0.14886304  0.1708091   0.09650978]
[ 0.26239728  0.0320204   0.00283366  0.20770757  0.          0.02238595
  0.          0.22470955  0.08047606  0.16746954]
```



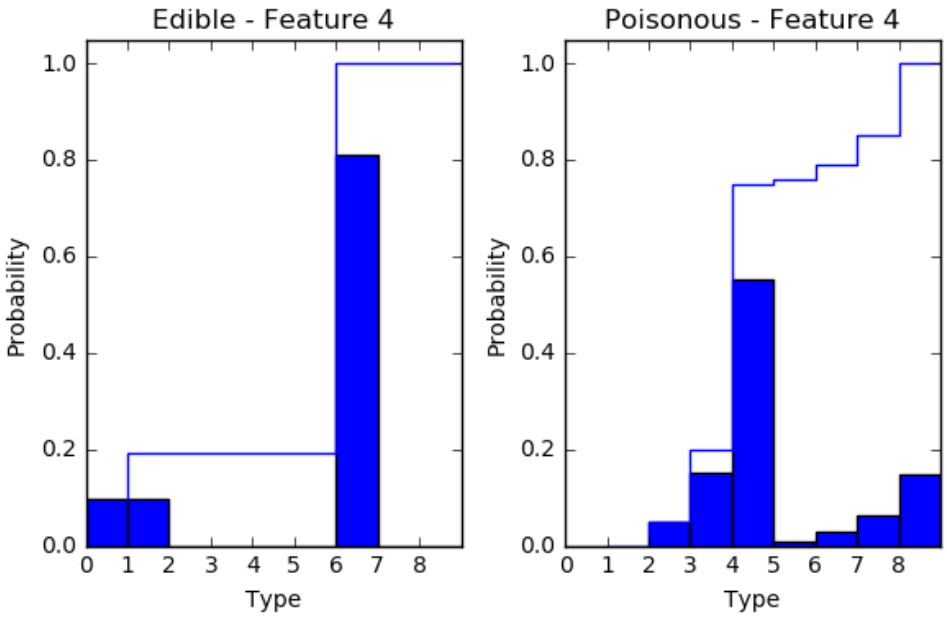



```
[ 0.65468006  0.34531994]
[ 0.15896855  0.84103145]
```






```
[ 0.09518773  0.09518773  0.          0.          0.          0.
  0.80962454  0.          0.        ]
[ 0.          0.          0.04817229  0.14990082  0.55171437  0.00878436
  0.0303202   0.06262397  0.14848399]
```


Edible - Feature 4
Probability
Type
Poisonous - Feature 4
Probability
Type

In [6]:
```
# naive bayes classification

#apriori estimates
prob_p = p/train_size
prob_e = e/train_size

r, c = X_test.shape
#print r,c
e_est = np.zeros((r, c-1))
p_est = np.zeros((r, c-1))

for j in range(1, 23):
    types = len(params[j-1])
    e_feature = np.zeros(types)
    p_feature = np.zeros(types)

    for v in range(types):
        e_feature[v] = ((X[:, 0] ==  1) & (X[:, j] == v+1)).sum()/e
        p_feature[v] = ((X[:, 0] == -1) & (X[:, j] == v+1)).sum()/p

    #print e_feature
    #print p_feature

    for i, x in enumerate(X_test):
        idx = int(x[j])
        e_est[i,j-1] = float(e_feature[idx-1])
        p_est[i,j-1] = float(p_feature[idx-1])

y_map = np.zeros(X_test.shape[0])
for i in xrange(r):
    if prob_p*np.prod(p_est[i,:]) >= prob_e*np.prod(e_est[i,:]):
        y_map[i] = -1
    else:
        y_map[i] = 1
#print y_map
#print X_test[:,0]
correct = ((y_map == X_test[:, 0].astype(int)).sum())
print correct, r
print "Accuracy: ", float(correct)/r
```

```
811 813
Accuracy:  0.9975399754
```