In [ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
```

In [ ]:

```
df = pd.read_csv("mushrooms.csv")
df.head()
print(df.shape)
```

```
(8124, 23)
```

In [ ]:

```
le = LabelEncoder()
ds = df.apply(le.fit_transform)
```

In [ ]:

```
ds.head()
```

Out[ ]:

	type	cap_shape	cap_surface	cap_color	bruises	odor	gill_attachment	gill_spacing	gill_size	gill_c
0	1	5	2	4	1	6	1	0	1	
1	0	5	2	9	1	0	1	0	0	
2	0	0	2	8	1	3	1	0	0	
3	1	5	3	8	1	6	1	0	1	
4	0	5	2	3	0	5	1	1	0	

5 rows × 23 columns

In [ ]:

```
data = ds.values
print(data.shape)
print(type(data))
print(data[:5,:])

data_x = data[:,1:]
data_y = data[:,0]
```

```
(8124, 23)
<class 'numpy.ndarray'>
[[1 5 2 4 1 6 1 0 1 4 0 3 2 2 7 7 0 2 1 4 2 3 5]
 [0 5 2 9 1 0 1 0 0 4 0 2 2 2 7 7 0 2 1 4 3 2 1]
 [0 0 2 8 1 3 1 0 0 5 0 2 2 2 7 7 0 2 1 4 3 2 3]
 [1 5 3 8 1 6 1 0 1 5 0 3 2 2 7 7 0 2 1 4 2 3 5]
 [0 5 2 3 0 5 1 1 0 4 1 3 2 2 7 7 0 2 1 0 3 0 1]]
```

In [ ]:

```
x_train,x_test,y_train,y_test = train_test_split(data_x,data_y,test_size=0.2)
```

In [ ]:

```
print(x_train.shape,y_train.shape)
print(x_test.shape,y_test.shape)
```

```
(6499, 22) (6499,)
(1625, 22) (1625,)
```

In [ ]:

```
def prior_prob(y_train,label):
    total_examples = y_train.shape[0]
    count_examples = np.sum(y_train==label)

    return (count_examples)/float(total_examples)
```

In [ ]:

```
def cond_prob(x_train,y_train,feature_col,feature_val,label):
    x_filtered = x_train[y_train==label]
    numerator = np.sum(x_filtered[:,feature_col]==feature_val)
    denominator = np.sum(y_train==label)

    return numerator/float(denominator)
```

In [ ]:

```
def predict(x_train,y_train,xtest):
    """xtest is a single testing point, n features """

    classes = np.unique(y_train)
    n_features = x_train.shape[1]
    post_probs = []

    for label in classes:

        likelihood = 1.0

        for f in range(n_features):
            cond = cond_prob(x_train,y_train,f,xtest[f],label)
            likelihood *=cond

        prior = prior_prob(y_train,label)
        post = likelihood*prior
        post_probs.append(post)

    pred = np.argmax(post_probs)
    return pred
```

In [ ]:

```
def score(x_train,y_train,x_test,y_test):
    pred = []

    for i in range(x_test.shape[0]):
        pred_label = predict(x_train,y_train,x_test[i])
        pred.append(pred_label)

    pred = np.array(pred)

    accuracy = np.sum(pred==y_test)/y_test.shape[0]
    return accuracy
```

In [ ]:

```
print(score(x_train,y_train,x_test,y_test))
```

```
0.9993846153846154
```

In [ ]:

```
```