In [28]:
```
import numpy as np
import matplotlib.pyplot as plt
import os.path
import pandas as pd
import scipy
# Load data
npz = np.load('heart_disease_data.npz')

df = pd.DataFrame.from_dict({item: npz[item] for item in npz.files}, orient='

train = df.iloc[0:50,:]

test = df.iloc[50:,:]

def ind_x_eq_val(x, val):
    return np.where(x==val)[0]

def count_x_eq_val(x, val):
    return len(ind_x_eq_val(x, val))/float(len(x))

def gaussian(x, mu, sig):
    return np.exp(-np.power(x - mu, 2.) / (2 * np.power(sig, 2.))) / sig / np
```

QUESTION (b)

Non-parametric model Compute empirical pmf, derive the conditional pmf, and estimate the MAP decision by the mode of posterior distribution $p_{\tilde{h}|\tilde{s},\tilde{c}}$. (The MAP estimates should be $\tilde{h} = 0 \text{ or } 1$)

In [41]:
```
# Estimate the pmf of H, i.e. P(H=0) and P(H=1)
P_H = np.zeros(2)

for i in range(2):
    P_H[i] = count_x_eq_val(df['heart_disease'],i)
    if i == 0:
        P_H0 = count_x_eq_val(df['heart_disease'],i)
    else:
        P_H1 = count_x_eq_val(df['heart_disease'],i)

# Estimate the conditional pmf of S given H, i.e. P(S|H=0) and P(S|H=1)
P_S_H0 = np.zeros(2)
P_S_H1 = np.zeros(2)
for ind_S in range(2):
    P_S_H0[ind_S] = len(df[(df['heart_disease'] == 0) & (df['sex']==ind_S)])
    P_S_H1[ind_S] = len(df[(df['heart_disease'] == 1) & (df['sex']==ind_S)])

# Estimate the conditional pmf of C given H, i.e. P(C|H=0) and P(C|H=1)
P_C_H0 = np.zeros(4)
P_C_H1 = np.zeros(4)
for ind_C in range(4):
    P_C_H0[ind_C] = len(df[(df['heart_disease'] == 0) & (df['chest_pain']==ir
    P_C_H1[ind_C] = len(df[(df['heart_disease'] == 1) & (df['chest_pain']==ir

h0_list = []
for s in range(2):
    for c in range(4):
        top = P_H[0]*P_S_H0[s]*P_C_H0[c]
        h0_list.append(top)

h1_list = []
for s in range(2):
    for c in range(4):
        top = P_H[1]*P_S_H1[s]*P_C_H1[c]
        h1_list.append(top)

map_estimate_S_C = []

for i in range(len(h0_list)):
    if h1_list[i] < h0_list[i]:
        map_estimate_S_C.append(0)
    else:
        map_estimate_S_C.append(1)

errors = 0
index = 0
df = df.iloc[0:50,:]
for s in range(2):
    for c in range(4):
        df_of_interest = df[(df['sex_test']==s) & (df['chest_pain_test']==c)]
        for i in df_of_interest['heart_disease_test']:
            if i != map_estimate_S_C[index]:
                errors += 1
        index += 1

error_rate_S_C = errors / len(df)
```

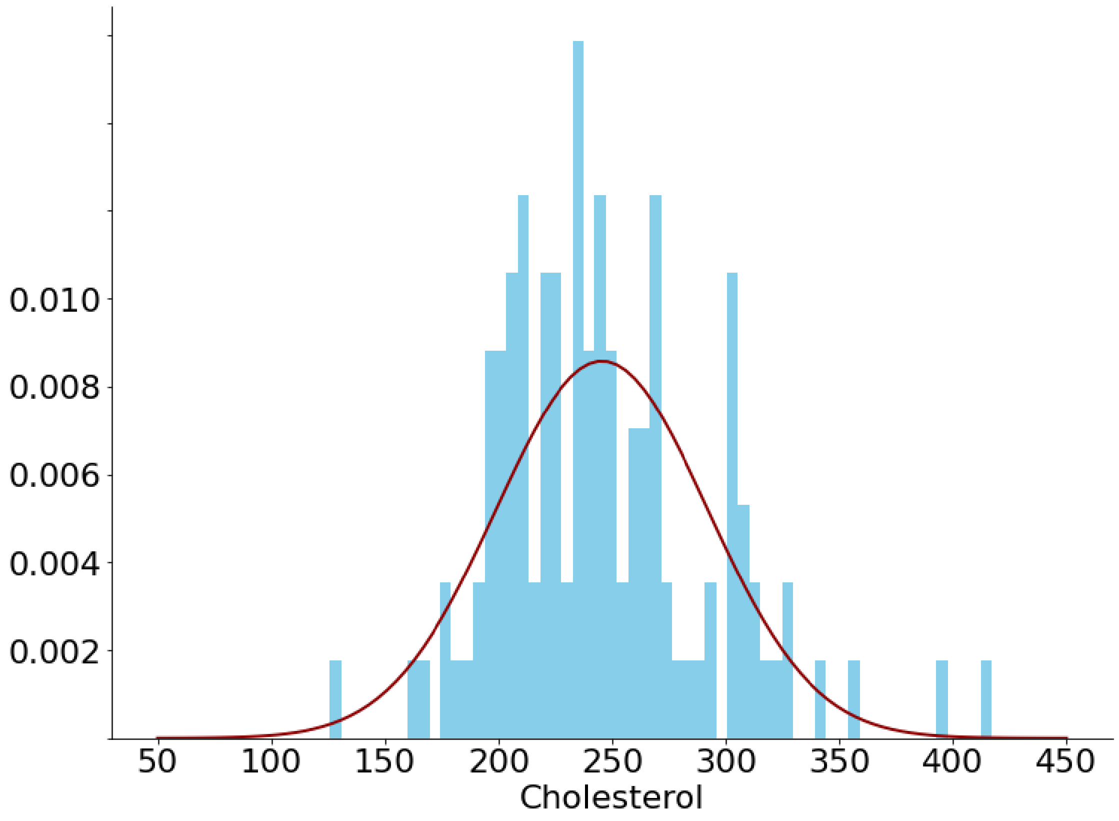
```
print("Probability of error " + str(error_rate_S_C))#
```

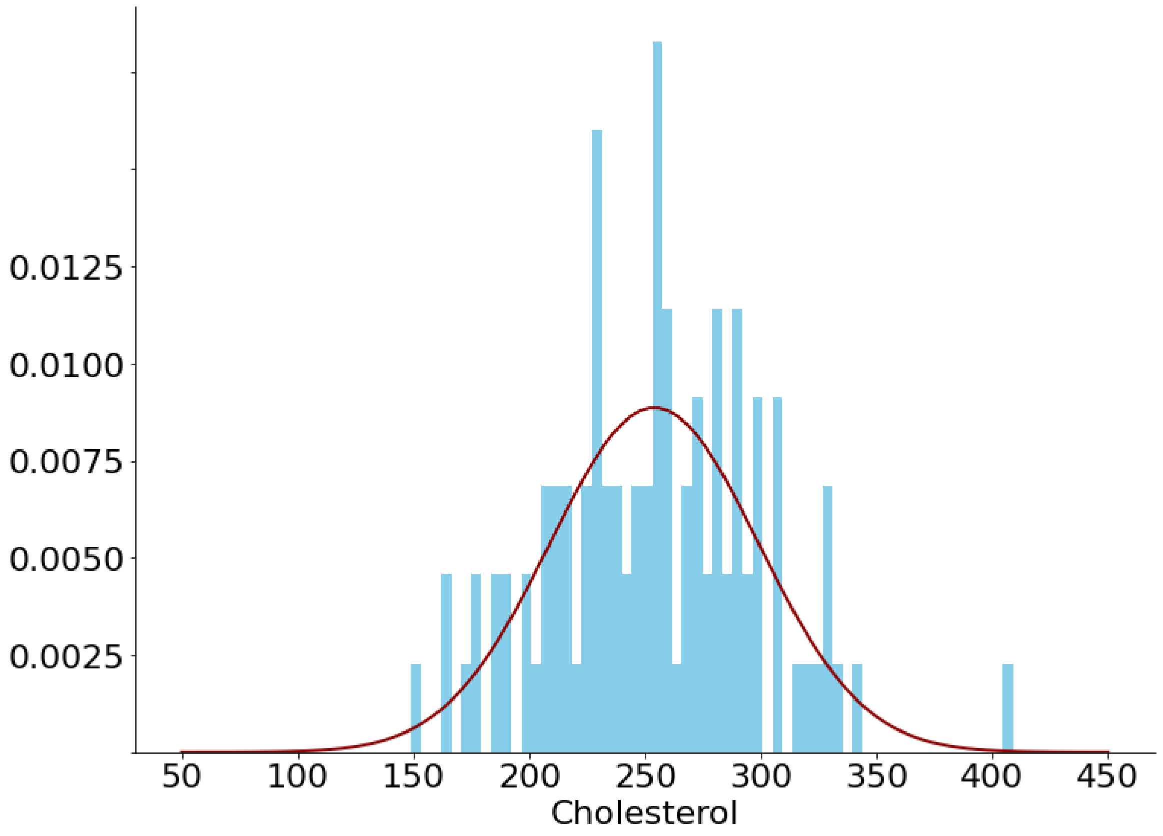
```
Probability of error 0.18
```

QUESTION (d)

Maximum likelihood estimates Find the parameters of two normal distributions ($\tilde{x}|\tilde{h} = 1$ and $\tilde{x}|\tilde{h} = 0$) that maximize the likelihood functions.

In [42]:
```
df = pd.DataFrame.from_dict({item: npz[item] for item in npz.files}, orient='

## Estimate MLE of X given H
no_heart_cond = (df[df['heart_disease']==0])
heart_cond = (df[df['heart_disease']==1])
mean_X_H = np.zeros(2)
std_X_H = np.zeros(2)
mean_X_H[0]= np.mean(np.array(no_heart_cond['cholesterol']))
std_X_H[0] = np.std(np.array(no_heart_cond['cholesterol']))
mean_X_H[1]= np.mean(np.array(heart_cond['cholesterol']))
std_X_H[1] = np.std(np.array(heart_cond['cholesterol']))

n_plot = 100
for i in range(2):
    plt.figure(figsize=(12, 9))
    ax = plt.subplot(111)
    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)
    ax.get_xaxis().tick_bottom()
    ax.get_yaxis().tick_left()
    yticks = ax.yaxis.get_major_ticks()
    yticks[0].label1.set_visible(False)
    plt.xticks(fontsize=22)
    plt.yticks(fontsize=22)
    plt.xlabel("Cholesterol", fontsize=22)

    plt.hist(df.loc[df['heart_disease']==i]['cholesterol'],
             bins=60,stacked=True,edgecolor = "none", color="skyblue", densit

    plt.plot(np.linspace(50, 450, n_plot),gaussian(np.linspace(50, 450, n_plc
                     mean_X_H[i], std_X_H[i]), color="darkred", lw=2)
```





QUESTION(e)

MAP decision compute posterior $p_{\tilde{h}|\tilde{s},\tilde{c},\tilde{x}}$ and derive MAP

In [43]:
```
from scipy import stats
```

In [44]:
```
# Calculate the MAP estimate
MAP_list = []
for s in range(2):
    for c in range(4):
        for chol in set(df['cholesterol_test']):
            pdf_value_H0 = scipy.stats.norm.pdf(chol, mean_X_H[0], std_X_H[0]
            pdf_value_H1 = scipy.stats.norm.pdf(chol, mean_X_H[1], std_X_H[1]
            var1 = P_H1*P_S_H1[s]*P_C_H1[c]*pdf_value_H1
            var2 = P_H0*P_S_H0[s]*P_C_H0[c]*pdf_value_H0
            if var1 < var2: MAP_list.append(0)
            else: MAP_list.append(1)


MAP_estimate_S_C_X = max(set(MAP_list), key=MAP_list.count)

# Calculate the error rate
list_index = 0
error_count = 0
for s in range(2):
    for c in range(4):
        for chol in set(df['cholesterol_test']):
            sub_df = train[(train['sex_test'] == s) & (train['chest_pain_test
            for idx2 in sub_df['heart_disease_test']:
                if idx2 != MAP_list[list_index]:
                    error_count += 1
            list_index += 1

error_rate_S_C_X = error_count / len(train)
print("Probability of error using cholesterol " + str(error_rate_S_C_X))
```

```
Probability of error using cholesterol 0.14
```

In [ ]:

In [ ]:

In [ ]: