Problem 2

[1]:
```
import numpy as np
spambase = np.loadtxt('spambase.data', delimiter=',')
```

[2]:
```
np.random.shuffle(spambase)
y = spambase[:, -1].astype(int)
X = np.zeros(np.array(spambase.shape) - np.array([0, 1]), dtype=int)
X[spambase[:, :-1] > np.median(spambase[:, :-1], axis=0)[None, :]] = 1
```

[3]:
```
X_train, y_train = X[:2000, :], y[:2000]
X_val, y_val = X[2000:, :], y[2000:]
```

[4]:
```
theta = np.array([
    X_train[np.logical_not(y_train), :].sum(axis=0) / np.logical_not(y_train).
 ↪sum(),
    # P[xi = 1 | y = 0] over i= 1,...,d
    X_train[y_train.astype(bool), :].sum(axis=0) / y_train.sum()
    # P[xi = 1 | y = 1] over i= 1,...,d
]).T
not_theta = 1 - theta
# P[xi = 0 | y = 0], P[xi = 0 | y = 1]
```

[5]:
```
eta = y_train.sum() / 2000  # P[y = 1]
not_eta = 1 - eta           # P[y = 0]
def y_hat(x):
    P = np.array([
        not_eta * theta[x.astype(bool), 0].prod() * not_theta[np.
 ↪logical_not(x), 0].prod(),
        eta * theta[x.astype(bool), 1].prod() * not_theta[np.logical_not(x), 1].
 ↪prod()
    ])  # P[y] * prod(P[xi | y]) over y = 0, 1
    return P.argmax()
```

[6]:
```
y_test = np.array([ y_hat(row) for row in X_val ])
error = np.abs(y_val - y_test).mean()
error
```

[6]:
```
0.1122645136485967
```

[7]:
```
y_naiv = int(y_train.mean() > 0.5)
error_naiv = np.abs(y_val - y_naiv).mean()
error_naiv
```

[7]:
```
0.405997693194925
```