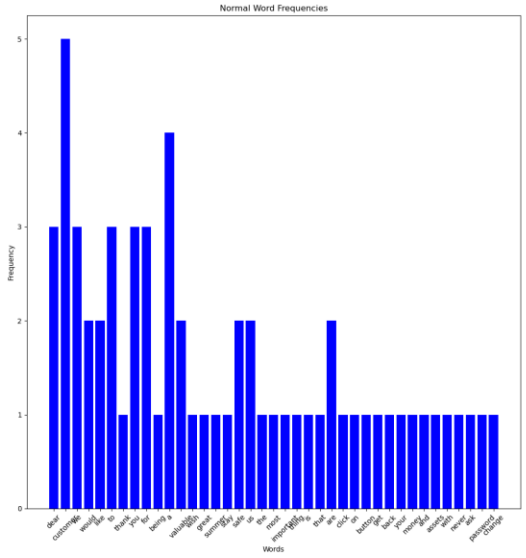
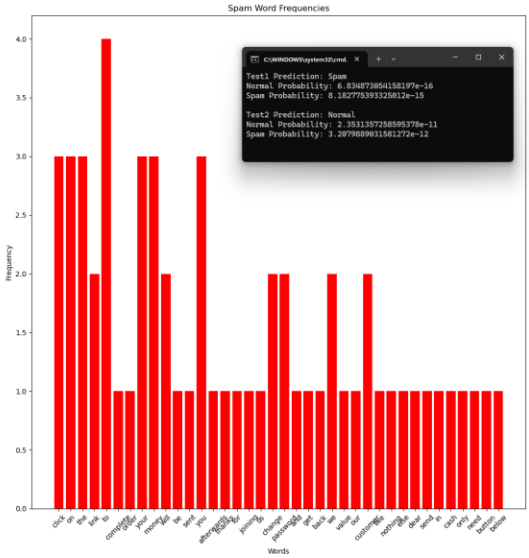
1.

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.decomposition import PCA
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from collections import Counter

#only works for binomial
def NBbyhand(Class0Dic, Class1Dic, prior0, prior1, Classnames, test, alpha=1):

    Class0total = sum(Class0Dic.values())
    Class1total = sum(Class1Dic.values())

    Class0prob = []
    Class1prob = []

    for word in test:
        Class0prob.append((Class0Dic.get(word, 0) + alpha) / (Class0total + alpha * len(Class0Dic)))
        Class1prob.append((Class1Dic.get(word, 0) + alpha) / (Class1total + alpha * len(Class1Dic)))

    #intialize prob with prior
    prob0 = prior0
    prob1 = prior1

    for value in Class0prob:
        prob0 *= value
    for value in Class1prob:
        prob1 *= value

    pred = 'NA'

    if prob0 > prob1:
        pred = Classnames[0]
    else:
        pred = Classnames[1]

    return (prob0, prob1, pred)


#Prior values
priorNorm = 0.73
priorSpam = 0.27

#Read files for plotting
with open("train_N.txt", "r") as f:
    train_N = f.read().split()

with open("train_S.txt", "r") as f:
    train_S = f.read().split()

#count the frequency of words
countsN = Counter(train_N)
countsS = Counter(train_S)

# Extract the words and their frequencies
key_listN = list(countsN.keys())
val_listN = list(countsN.values())

```

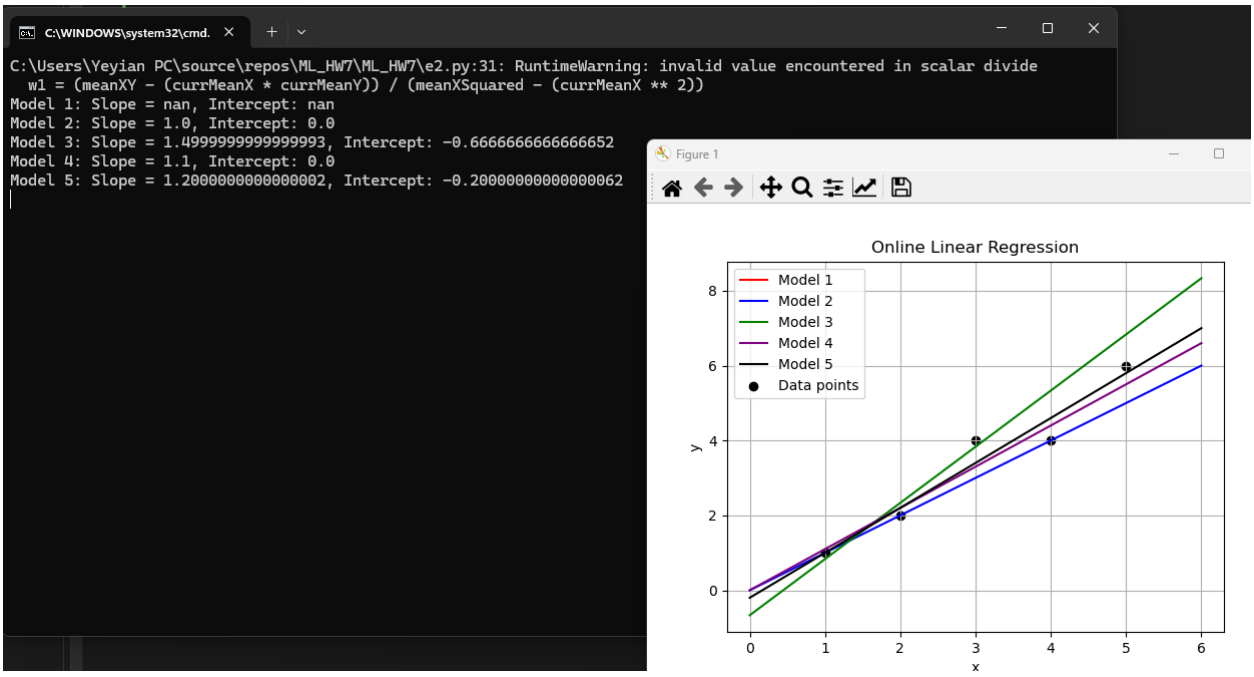
```
# read in test data
with open("testEmail_I.txt", "r") as f:
    test1 = f.read().split()

with open("testEmail_II.txt", "r") as f:
    test2 = f.read().split()

#my pred
Classnames = ['Normal', 'Spam']

#test1
prob0, prob1, pred =  NBbyhand(countsN, countsS, priorNorm, priorSpam, Classnames, test1)
print(f'Test1 Prediction: {pred}')
print(f'Normal Probability: {prob0}')
print(f'Spam Probability: {prob1}')

# test2
prob0, prob1, pred =  NBbyhand(countsN, countsS, priorNorm, priorSpam, Classnames, test2)
print()
print(f'Test2 Prediction: {pred}')
print(f'Normal Probability: {prob0}')
print(f'Spam Probability: {prob1}')


#plots
figure, axis = plt.subplots(1, 2)

# Plot for Normal Words
axis[0].bar(key_listN, val_listN, color='blue')
axis[0].set_title("Normal Word Frequencies")
axis[0].set_xlabel("Words")
axis[0].set_ylabel("Frequency")
axis[0].tick_params(axis='x', rotation=45)# way to nake the words more legible

#Plot for Spam Words
axis[1].bar(key_listS, val_listS, color='red')
axis[1].set_title("Spam Word Frequencies")
axis[1].set_xlabel("Words")
axis[1].set_ylabel("Frequency")
axis[1].tick_params(axis='x', rotation=45) #way to nake the words more legible

# hopfully better spacing and readability
plt.tight_layout()

plt.show()
```


Normal Word Frequencies
Frequency
Words


Spam Word Frequencies
Frequency
Words
C:\WINDOWS\system32\cmd.
Test1 Prediction: Spam
Normal Probability: 6.834873054158197e-16
Spam Probability: 8.182775393250812e-15
Test2 Prediction: Normal
Normal Probability: 2.353135725859537 8e-11
Spam Probability: 3.207988903158127 2e-12

2.

```python
import matplotlib.pyplot as plt
import numpy as np

np.set_printoptions(precision=2)

#data
x = np.array([1, 2, 3, 4, 5])
y = np.array([1, 2, 4, 4, 6])

# line spec for plotting
c = ['r', 'b', 'g', 'purple', 'black']
labels = ['Model 1', 'Model 2', 'Model 3', 'Model 4', 'Model 5']

#for plotting
xline = np.linspace(0, 6, 100)


# Ref slide 395 of python for data sci slides (least squares)
for i in range(len(x)):

    #calculate current mean
    currMeanX = np.mean(x[:i+ 1])
    currMeanY = np.mean(y[:i+ 1])

    #calculate mean of X*Y
    meanXY = np.mean(x[:i+ 1] * y[:i+ 1])
    #calculate mean of X squared
    meanXSquared = np.mean(x[:i+ 1] ** 2)

    #calculate intercept and coeff
    w1 = (meanXY - (currMeanX * currMeanY)) / (meanXSquared - (currMeanX ** 2))
    w0 = currMeanY - (w1 * currMeanX)

    #make model
    ymodel = w1 * xline + w0

    #print vals
    print(f'Model {i + 1}: Slope = {w1}, Intercept: {w0}')

    #plt current model
    plt.plot(xline, ymodel, color=c[i], label=labels[i])


# original data points
plt.scatter(x, y, color='black', label='Data points')

# Plot labels
plt.title('Online Linear Regression')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid()
plt.show()
```


C:\WINDOWS\system32\cmd.
C:\Users\Yeyian PC\source\repos\ML_HW7\ML_HW7\e2.py:31: RuntimeWarning: invalid value encountered in scalar divide
w1 = (meanXY - (currMeanX * currMeanY)) / (meanXSquared - (currMeanX ** 2))
Model 1: Slope = nan, Intercept: nan
Model 2: Slope = 1.0, Intercept: 0.0
Model 3: Slope = 1.4999999999999993, Intercept: -0.6666666666666652
Model 4: Slope = 1.1, Intercept: 0.0
Model 5: Slope = 1.2000000000000002, Intercept: -0.20000000000000062
Figure 1
Online Linear Regression
Model 1
Model 2
Model 3
Model 4
Model 5
Data points
y
x
0
1
2
3
4
5
6
8