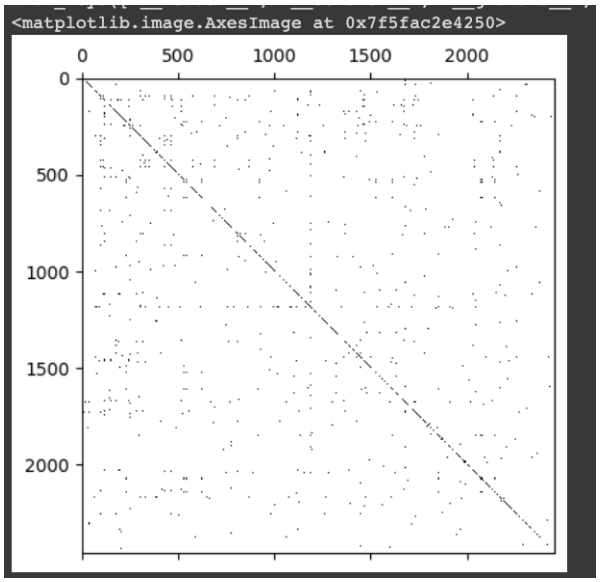
Zachary Anderson, Andrew Ng

Lab 13

2.1

```
gene_dataset_path = '/content/gdrive/MyDrive/ESE 224/lab13_data/geneNetwork_rawPCNCI.mat'
data = scipy.io.loadmat(gene_dataset_path)
print(data.keys())
adj_matrix = data['geneNetwork_rawPCNCI']
plt.spy(adj_matrix)
```

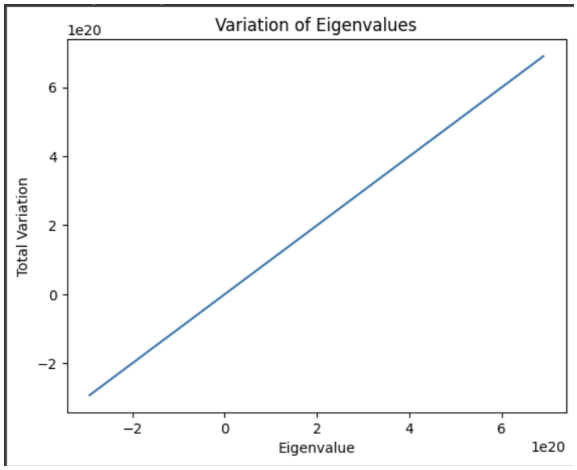
```
<matplotlib.image.AxesImage at 0x7f5fac2e4250>
```



Because there are values greater than zero along the diagonal of the adjacency matrix, there must be self-loops. Summing along the diagonal, we reveal 1525 self-loops in the graph. We know the graph is undirected and unweighted. This is due to the adjacency matrix being symmetric and containing values of only zero or one. If define another adjacency matrix A* (removing all self loops from A), the Laplacian of A* is the exact same as the Laplacian of A of A, as there is no change by removing the self-loops from A.

2.2

```
def totalVariation(x, S):
    return x.T@(S)@x

S = laplacian_matrix
[eigs,V]=np.linalg.eig(S)
V=V[:,np.argsort(eigs)]
eigs=np.sort(eigs)

variation = np.diag(V.T @ S @ V)
#print(abs(variation))
plt.plot(eigs, variation)
plt.title('Variation of Eigenvalues')
plt.xlabel('Eigenvalue')
plt.ylabel('Total Variation')
plt.show()
```



From our findings, we see that as the eigenvalues increase, our total variation of the signal increases as well. Eigenvectors associated with larger eigenvalues tend to oscillate faster as opposed to eigenvectors with smaller eigenvalues. We observe a linear relationship.

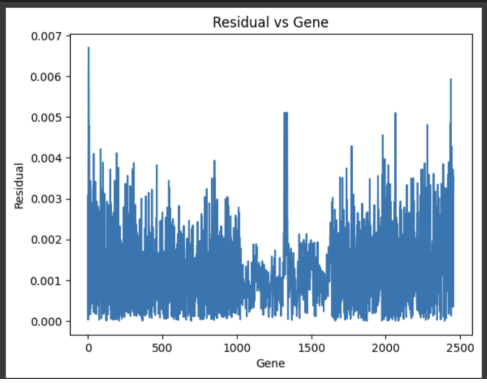
3.1

```
[16] def computeGFT(x, V, k=None):
        xt=np.conj(V.T) @ x
        if k==None:
          return xt
        else:
          xtk = np.zeros_like(xt)
          xtk[np.argsort(np.abs(xt[:, 0]))[-k:], 0] = xt[np.argsort(np.abs(xt[:, 0]))[-k:], 0]
          return xtk


    def laplacian(A):
      diagonal = []
      for i in range(240):
        degree = 0
        for j in A[i]:
          degree += j
        diagonal.append(degree)
      #print(diagonal)
      N = len(diagonal)
      L = np.zeros((N, N))
      for i in range(N):
        L[i, i] = diagonal[i]
      L = L - A
      return L
```

```
[17] V_hermitian = np.conj(V).transpose()
     GFT = np.matmul(V_hermitian, np.transpose(mutation_mtx))

     DP = np.zeros(2458)
     histology_subtype = hist_data['histology_subtype']
```

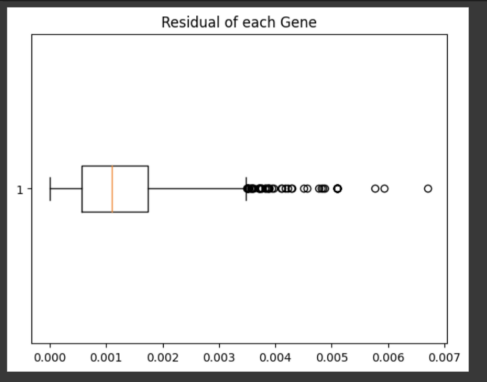
```
     ones = []
     twos = []
     for i in range(240):
       if histology_subtype[i] == 1:
         ones.append(i)
       else:
         twos.append(i)

     print(ones)
     num_ones = len(ones)
     num_twos = len(twos)
     for k in range(2458):
       sum_ones = 0
       sum_twos = 0
       total = 0
       for i in ones:
         #print(i)
         n = GFT[k,i]
         #print(n.shape)
         sum_ones = sum_ones + n
         total = total + np.abs(n)

       for j in twos:
         m = GFT[k,j]
         sum_twos = sum_twos + m
         total = total + np.abs(m)

       temp = np.abs((sum_ones/num_ones) - (sum_twos/num_twos))
       DP[k] = temp/total
```

```
indices = np.arange(len(DP))

plt.plot(indices, DP)
plt.xlabel('Gene')
plt.ylabel('Residual')
plt.title('Residual vs Gene')
plt.show()
```



3.2

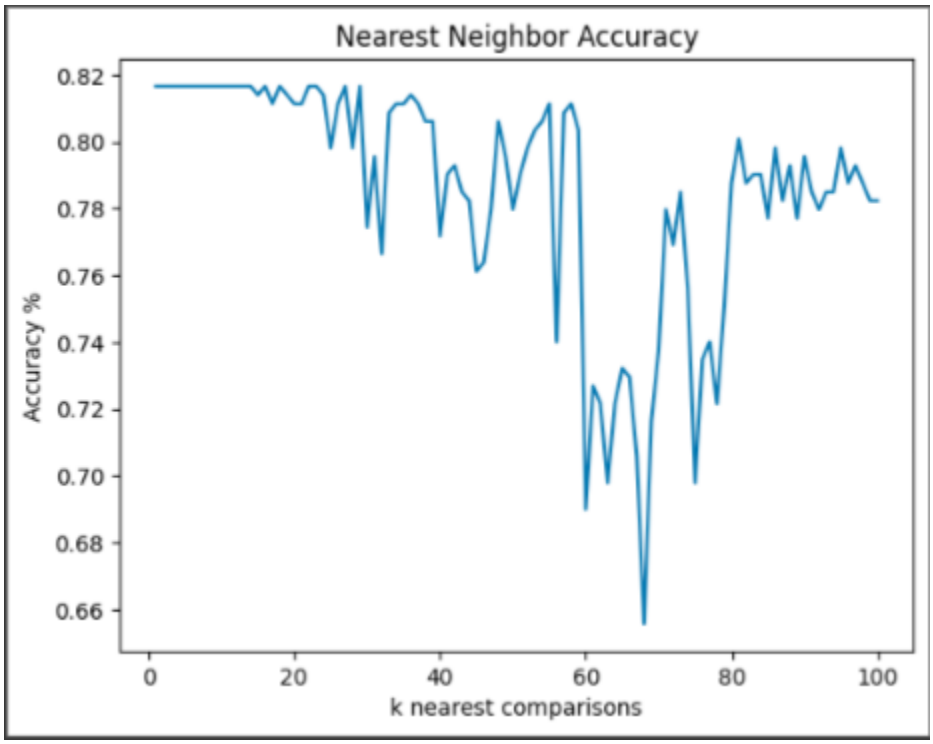
```
plt.boxplot(DP, vert = False)
plt.title('Residual of each Gene')
plt.show()
```



Our boxplot shows that the majority of the genes are not indicative of the phenotype. The outliers that do occur help determine what kind of cancer a patient might have. The gene with the highest magnitude is most likely indicative of the type of cancer.

4.1

```python
import numpy as np
from scipy.spatial.distance import pdist, squareform
from scipy import stats as st

def kNN(sig, type, max_k):
    pd = squareform(pdist(sig))

    k = np.arange(1, max_k+1)
    acc = np.zeros(max_k)

    for i in range(max_k):
        z = np.zeros(len(type))
        for j in range(len(type)):
            nn = np.argsort(pd[:,j])
            z[j] = (st.mode(type[nn[2::i+1],:])).mode
        acc[i] = np.mean(z == type)

    return k, acc
```



k=3: 81.67% Accuracy
k=5: 81.67% Accuracy
k=7: 81.67% Accuracy

4.2