**PCA applied to the data set to reduce it to a two dimensional set**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import pylab

from mpl\_toolkits.mplot3d import Axes3D

df = pd.read\_csv("data.csv",header = 0)

df= df.drop(labels = ['id'], axis=1)

df1=df[df.diagnosis == 'M']

df2=df[df.diagnosis == 'B']

df1= df1.drop(labels = ['diagnosis'], axis=1)

df2= df2.drop(labels = ['diagnosis'], axis=1)

from sklearn.decomposition import PCA

pca= PCA(n\_components=2)

pca.fit(df1)

PCA(copy=True, n\_components=2, whiten=False)

T=pca.transform(df1)

df1=pd.DataFrame(T)

df1.columns=['col1','col2']

pca= PCA(n\_components=2)

pca.fit(df2)

PCA(copy=True, n\_components=2, whiten=False)

T=pca.transform(df2)

df2=pd.DataFrame(T)

df2.columns=['col1','col2']

"""

fig=plt.figure()

ax=Axes3D(fig)

ax.scatter(df1.col1,df1.col2,df1.col3, c='r', label='malignant')

ax.scatter(df2.col1,df2.col2,df2.col3, c='b', label='benign')

plt.show()

"""

fig, ax = plt.subplots()

ax.scatter(df1.col1, df1.col2, c='r', label='malignant')

ax.scatter(df2.col1, df2.col2, c='b', label='benign')

plt.show()

print len(df1)

print len(df2)

df = pd.concat([df1,df2])

print len(df)

class\_label = []

for i in range(202):

class\_label.append('M')

for i in range(342):

class\_label.append('B')

df['diagnosis'] = class\_label

df.to\_csv("Compressed\_Data.csv")

**Nearest Neighbor tester for Malignant cases**

import pandas as pd

import math

df = pd.read\_csv("Compressed\_Data.csv")

df = df.drop(df.columns[0], axis=1)

df\_m = pd.read\_csv("Malignant.csv")

df\_m= df\_m.drop(labels = ['id','diagnosis'], axis=1)

from sklearn.decomposition import PCA

pca= PCA(n\_components=2)

pca.fit(df\_m)

PCA(copy=True, n\_components=2, whiten=False)

T=pca.transform(df\_m)

df\_m=pd.DataFrame(T)

df\_m.columns=['col1','col2']

dist\_list = []

for i in range(len(df\_m)):

new\_list = []

for j in range(len(df)):

x = math.sqrt((float(df.ix[j][0]) - float(df\_m.ix[i][0]))\*\*2 + (float(df.ix[j][1]) - float(df\_m.ix[i][1]))\*\*2)

new\_list.append(x)

dist\_list.append(new\_list)

correct = 0

for i in range(len(df\_m)):

x = min(dist\_list[i])

y = dist\_list[i].index(x)

if df.ix[y][2] == 'M':

correct += 1

print "accuracy is " + str(correct/10.0)

**Nearest Neighbor tester for Benign cases**

import pandas as pd

import math

df = pd.read\_csv("Compressed\_Data.csv")

df = df.drop(df.columns[0], axis=1)

df\_m = pd.read\_csv("Benign.csv")

df\_m= df\_m.drop(labels = ['id','diagnosis'], axis=1)

from sklearn.decomposition import PCA

pca= PCA(n\_components=2)

pca.fit(df\_m)

PCA(copy=True, n\_components=2, whiten=False)

T=pca.transform(df\_m)

df\_m=pd.DataFrame(T)

df\_m.columns=['col1','col2']

dist\_list = []

for i in range(len(df\_m)):

new\_list = []

for j in range(len(df)):

x = math.sqrt((float(df.ix[j][0]) - float(df\_m.ix[i][0]))\*\*2 + (float(df.ix[j][1]) - float(df\_m.ix[i][1]))\*\*2)

new\_list.append(x)

dist\_list.append(new\_list)

correct = 0

for i in range(len(df\_m)):

x = min(dist\_list[i])

y = dist\_list[i].index(x)

if df.ix[y][2] == 'B':

correct += 1

print "accuracy is " + str(correct/15.0)

**K Nearest Neighbor for Malignant cases**

import pandas as pd

import math

df = pd.read\_csv("Compressed\_Data.csv")

df = df.drop(df.columns[0], axis=1)

df\_m = pd.read\_csv("Malignant.csv")

df\_m= df\_m.drop(labels = ['id','diagnosis'], axis=1)

from sklearn.decomposition import PCA

pca= PCA(n\_components=2)

pca.fit(df\_m)

PCA(copy=True, n\_components=2, whiten=False)

T=pca.transform(df\_m)

df\_m=pd.DataFrame(T)

df\_m.columns=['col1','col2']

dist\_list = []

for i in range(len(df\_m)):

new\_list = []

for j in range(len(df)):

x = math.sqrt((float(df.ix[j][0]) - float(df\_m.ix[i][0]))\*\*2 + (float(df.ix[j][1]) - float(df\_m.ix[i][1]))\*\*2)

new\_list.append(x)

dist\_list.append(new\_list)

k\_list = []

mint = 2

maxt = 30

for l in range(mint,maxt):

correct = 0

for i in range(len(df\_m)):

list1 = list(dist\_list[i])

list1.sort()

list1 = list(list1[0:l])

list2 = []

for j in range(len(list1)):

y = dist\_list[i].index(list1[j])

c = df.ix[y][2]

list2.append(c)

M\_count = 0

B\_count = 0

for k in range(len(list2)):

if list2[k] == 'M':

M\_count += 1

else:

B\_count += 1

if M\_count > B\_count:

correct += 1

print "Accuracy for " + str(l) + " neighbors is - " + str(100\*correct/10.0) + " percent"

k\_list.append(correct/10.0)

maxi = max(k\_list)

ind = k\_list.index(maxi)

print "\nMax accuracy with " + str(ind+mint) + " neighbors"

print "Accuracy rate obtained - " + str(100 \* maxi) + " percent"

**K Nearest Neighbor for Benign cases**

import pandas as pd

import math

df = pd.read\_csv("Compressed\_Data.csv")

df = df.drop(df.columns[0], axis=1)

df\_m = pd.read\_csv("Benign.csv")

df\_m= df\_m.drop(labels = ['id','diagnosis'], axis=1)

from sklearn.decomposition import PCA

pca= PCA(n\_components=2)

pca.fit(df\_m)

PCA(copy=True, n\_components=2, whiten=False)

T=pca.transform(df\_m)

df\_m=pd.DataFrame(T)

df\_m.columns=['col1','col2']

dist\_list = []

for i in range(len(df\_m)):

new\_list = []

for j in range(len(df)):

x = math.sqrt((float(df.ix[j][0]) - float(df\_m.ix[i][0]))\*\*2 + (float(df.ix[j][1]) - float(df\_m.ix[i][1]))\*\*2)

new\_list.append(x)

dist\_list.append(new\_list)

k\_list = []

mint = 2

maxt = 30

for l in range(mint,maxt):

correct = 0

for i in range(len(df\_m)):

list1 = list(dist\_list[i])

list1.sort()

list1 = list(list1[0:l])

list2 = []

for j in range(len(list1)):

y = dist\_list[i].index(list1[j])

c = df.ix[y][2]

list2.append(c)

M\_count = 0

B\_count = 0

for k in range(len(list2)):

if list2[k] == 'B':

M\_count += 1

else:

B\_count += 1

if M\_count > B\_count:

correct += 1

print "Accuracy for " + str(l) + " neighbors is - " + str(100\*correct/15.0) + " percent"

k\_list.append(correct/15.0)

maxi = max(k\_list)

ind = k\_list.index(maxi)

print "\nMax accuracy with " + str(ind+mint) + " neighbors"

print "Accuracy rate obtained - " + str(maxi \* 100) + " percent"

**Logistic Regression**

import pandas as pd

import math

df = pd.read\_csv("Compressed\_Data.csv")

df = df.drop(df.columns[0], axis=1)

df.col1 = (df.col1 - df.col1.mean())/df.col1.std(ddof=0)

df.col2 = (df.col2 - df.col2.mean())/df.col2.std(ddof=0)

m = len(df)

alpha=0.5

theta = [0.059934091281950624, 0.66013667247412766, 2.2240077122167441, 5.200267436045749, -0.34186706640479331, 0.72338346402593545, -0.51447600601326748, 1.1963604736003937, 0.9451943759502669, -3.0331260569314291, 0.11419621609911793, 0.061088064230656512, 0.42762709404695665, 0.12808445207127342, 0.089804995954129385]

cost = 1720.0

prev\_cost = 1721.0

new\_list = []

for i in range(len(df)):

new\_list.append(0.0)

df1 = pd.DataFrame()

df1['col1'] = new\_list

df1['diagnosis'] = df['diagnosis']

while cost > 40.0 and cost < prev\_cost:

print "Theta vector is - " + str(theta)

for i in range(len(df)):

z = theta[0] \* df.ix[i][0] + theta[1] \* df.ix[i][1] + theta[2] \* df.ix[i][0] \*\* 2 + theta[3] \* df.ix[i][1] \*\* 2 + theta[4] \* df.ix[i][0] \* df.ix[i][1]

z += theta[5] \* df.ix[i][0] \*\* 3 + theta[6] \* df.ix[i][0] \*\* 2 \* df.ix[i][1] + theta[7] \* df.ix[i][0] \* df.ix[i][1] \*\* 2 + theta[8] \* df.ix[i][1] \*\* 3 + theta[9]

z += theta[10] \* df.ix[i][0] \*\* 4 + theta[11] \* df.ix[i][0] \*\* 3 \* df.ix[i][1] + theta[12] \* df.ix[i][0] \*\* 2 \* df.ix[i][1] \*\* 2 + theta[13] \* df.ix[i][0] \* df.ix[i][1] \*\* 3 + theta[14] \* df.ix[i][1] \*\* 4

hx = 1/(1+math.exp(-z))

df1.set\_value(i,'col1',hx)

prev\_cost = cost

cost = 0

for j in range(len(df1)):

if df1.ix[j][1] == 'M':

y = 1

else:

y=0

cost += (-y\*math.log(df1.ix[j][0],10) - ((1-y)\*math.log((1.00001-df1.ix[j][0]),10)))

print "Cost is : " + str(cost)

J = -(1/float(m))\*cost

print "J value is : " + str(J)

grad = [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]

for i in range(len(df1)):

if df1.ix[i][1] == 'M':

y = 1

else:

y=0

grad[9] += df1.ix[i][0] - y

grad[0] += (df1.ix[i][0] - y) \* df.ix[i][0]

grad[1] += (df1.ix[i][0] - y) \* df.ix[i][1]

grad[2] += (df1.ix[i][0] - y) \* df.ix[i][0] \*\* 2

grad[3] += (df1.ix[i][0] - y) \* df.ix[i][1] \*\* 2

grad[4] += (df1.ix[i][0] - y) \* df.ix[i][0] \* df.ix[i][1]

grad[5] += (df1.ix[i][0] - y) \* df.ix[i][0] \*\* 3

grad[6] += (df1.ix[i][0] - y) \* df.ix[i][0] \*\* 2 \* df.ix[i][1]

grad[7] += (df1.ix[i][0] - y) \* df.ix[i][0] \* df.ix[i][1] \*\* 2

grad[8] += (df1.ix[i][0] - y) \* df.ix[i][1] \*\* 3

grad[10] += (df1.ix[i][0] - y) \* df.ix[i][0] \*\* 4

grad[11] += (df1.ix[i][0] - y) \* df.ix[i][0] \*\* 3 \* df.ix[i][1]

grad[12] += (df1.ix[i][0] - y) \* df.ix[i][0] \*\* 2 \* df.ix[i][1] \*\* 2

grad[13] += (df1.ix[i][0] - y) \* df.ix[i][0] \* df.ix[i][1] \*\* 3

grad[14] += (df1.ix[i][0] - y) \* df.ix[i][1] \*\* 4

for i in range(len(theta)):

theta[i] = theta[i] - (alpha \* grad[i] \* (1/float(m)))

print theta

**Logistic Regression for Malignant cases**

import pandas as pd

import math

df1 = pd.read\_csv("Malignant.csv")

df1 = df1.drop(labels = ['id','diagnosis'], axis=1)

df2 = pd.read\_csv("Benign.csv")

df2 = df2.drop(labels = ['id','diagnosis'], axis=1)

df = pd.concat([df1,df2])

from sklearn.decomposition import PCA

pca= PCA(n\_components=2)

pca.fit(df)

T=pca.transform(df)

df=pd.DataFrame(T)

df.columns=['col1','col2']

df.col1 = (df.col1 - df.col1.mean())/df.col1.std(ddof=0)

df.col2 = (df.col2 - df.col2.mean())/df.col2.std(ddof=0)

df = df[0:10]

theta = [0.059934091281950624, 0.66013667247412766, 2.2240077122167441, 5.200267436045749, -0.34186706640479331, 0.72338346402593545, -0.51447600601326748, 1.1963604736003937, 0.9451943759502669, -3.0331260569314291, 0.11419621609911793, 0.061088064230656512, 0.42762709404695665, 0.12808445207127342, 0.089804995954129385]

correct = 0

for i in range(len(df)):

hx = theta[0] \* df.ix[i][0] + theta[1] \* df.ix[i][1] + theta[2] \* df.ix[i][0] \*\* 2 + theta[3] \* df.ix[i][1] \*\* 2 + theta[4] \* df.ix[i][0] \* df.ix[i][1]

hx += theta[5] \* df.ix[i][0] \*\* 3 + theta[6] \* df.ix[i][0] \*\* 2 \* df.ix[i][1] + theta[7] \* df.ix[i][0] \* df.ix[i][1] \*\* 2 + theta[8] \* df.ix[i][1] \*\* 3 + theta[9]

hx += theta[10] \* df.ix[i][0] \*\* 4 + theta[11] \* df.ix[i][0] \*\* 3 \* df.ix[i][1] + theta[12] \* df.ix[i][0] \*\* 2 \* df.ix[i][1] \*\* 2 + theta[13] \* df.ix[i][0] \* df.ix[i][1] \*\* 3 + theta[14] \* df.ix[i][1] \*\* 4

hx = 1/(1+math.exp(-hx))

print hx

if hx >= 0.5:

correct += 1

print "Accuracy is : " + str(correct \* 100.0/10.0) + " percent"

**Logistic Regression for Benign cases**

import pandas as pd

import math

df1 = pd.read\_csv("Benign.csv")

df1 = df1.drop(labels = ['id','diagnosis'], axis=1)

df2 = pd.read\_csv("Malignant.csv")

df2 = df2.drop(labels = ['id','diagnosis'], axis=1)

df = pd.concat([df1,df2])

from sklearn.decomposition import PCA

pca= PCA(n\_components=2)

pca.fit(df)

T=pca.transform(df)

df=pd.DataFrame(T)

df.columns=['col1','col2']

df.col1 = (df.col1 - df.col1.mean())/df.col1.std(ddof=0)

df.col2 = (df.col2 - df.col2.mean())/df.col2.std(ddof=0)

df = df[0:15]

theta = [0.059934091281950624, 0.66013667247412766, 2.2240077122167441, 5.200267436045749, -0.34186706640479331, 0.72338346402593545, -0.51447600601326748, 1.1963604736003937, 0.9451943759502669, -3.0331260569314291, 0.11419621609911793, 0.061088064230656512, 0.42762709404695665, 0.12808445207127342, 0.089804995954129385]

correct = 0

for i in range(len(df)):

hx = theta[0] \* df.ix[i][0] + theta[1] \* df.ix[i][1] + theta[2] \* df.ix[i][0] \*\* 2 + theta[3] \* df.ix[i][1] \*\* 2 + theta[4] \* df.ix[i][0] \* df.ix[i][1]

hx += theta[5] \* df.ix[i][0] \*\* 3 + theta[6] \* df.ix[i][0] \*\* 2 \* df.ix[i][1] + theta[7] \* df.ix[i][0] \* df.ix[i][1] \*\* 2 + theta[8] \* df.ix[i][1] \*\* 3 + theta[9]

hx += theta[10] \* df.ix[i][0] \*\* 4 + theta[11] \* df.ix[i][0] \*\* 3 \* df.ix[i][1] + theta[12] \* df.ix[i][0] \*\* 2 \* df.ix[i][1] \*\* 2 + theta[13] \* df.ix[i][0] \* df.ix[i][1] \*\* 3 + theta[14] \* df.ix[i][1] \*\* 4

hx = 1/(1+math.exp(-hx))

print hx

if hx <= 0.5:

correct += 1

print "Accuracy is : " + str(correct \* 100.0/15.0) + " percent"