Programming Questions

Ali Tejani, amt3639; Caroline Yao, chy253

In [81]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

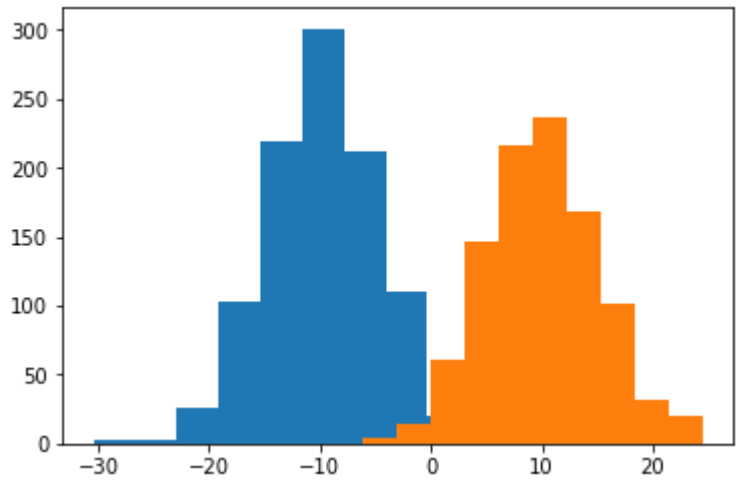
Problem 1

(a)

In [82]:
```
# take 1000 samples from gaussian distribution with mean -10 and standard deviation 5
gaussian1 = np.random.normal(-10,5,1000)
# take 1000 samples from gaussian distribution with mean 10 and standard deviation 5
gaussian2 = np.random.normal(10,5,1000)
```

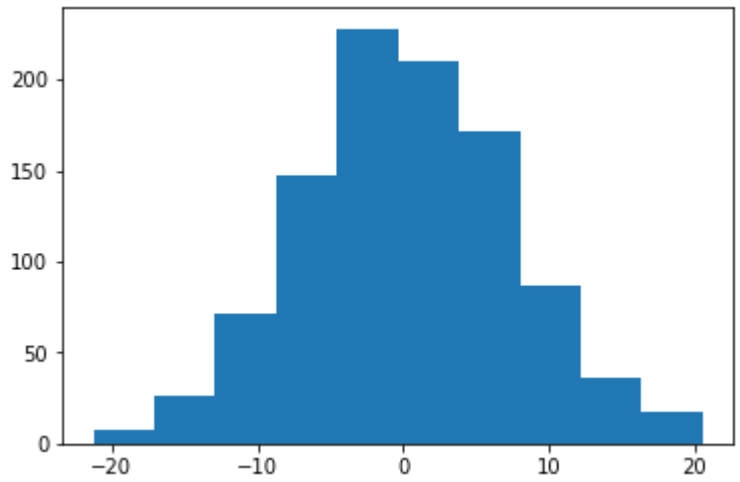
In [83]:
```
# plot histograms for both sets of points
plt.hist(gaussian1)
plt.hist(gaussian2)
```

Out[83]:
```
(array([   4.,   14.,   61.,  146.,  217.,  237.,  168.,  102.,   31.,   20.]),
 array([ -6.19971152,  -3.12920069,  -0.05868986,   3.01182097,
          6.08233181,   9.15284264,  12.22335347,  15.2938643 ,
         18.36437514,  21.43488597,  24.5053968 ]),
 <a list of 10 Patch objects>)
```



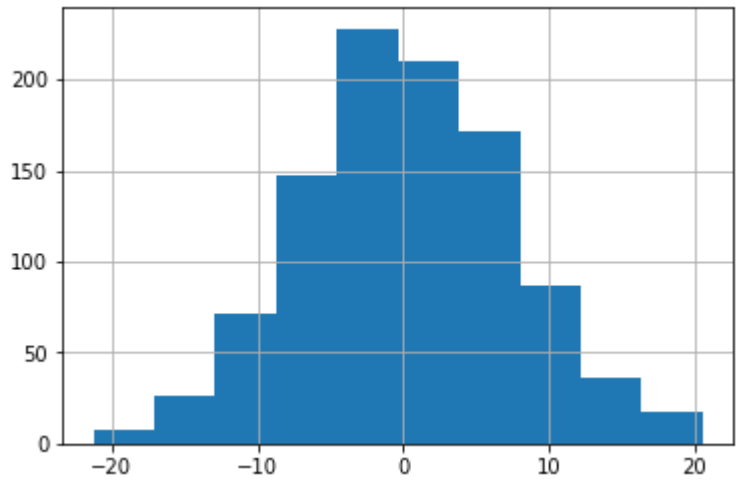
In [84]:
```
# add two sets together, point by point and plot histogram
sum1 = gaussian1 + gaussian2
plt.hist(sum1)
```

Out[84]:
```
(array([   7.,   26.,   71.,  147.,  228.,  210.,  171.,   87.,   36.,   1
7.]),
 array([-21.27018961, -17.08870724, -12.90722487,  -8.7257425 ,
        -4.54426013,  -0.36277776,   3.8187046 ,   8.00018697,
        12.18166934,  16.36315171,  20.54463408]),
 <a list of 10 Patch objects>)
```



In [85]:
```
sumSeries = pd.Series(sum1)
sumSeries.hist()
```

Out[85]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xddcc518>
```



The graph above shows that adding two gaussian distributions results in a new gaussian curve.

(b)

In [86]:
```
# mean of set
sumSeries.mean()
```

Out[86]:
```
0.22518396773554147
```

In [87]:
```
# variance of set
var = sumSeries.var()
var
```

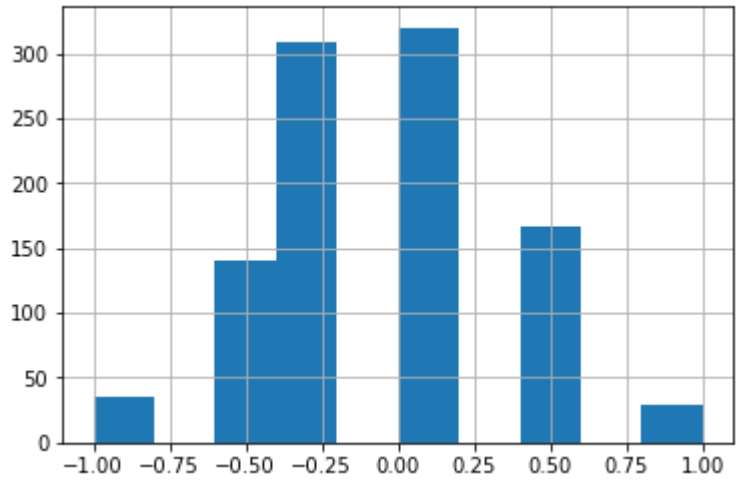
Out[87]:
```
51.98546320738543
```

The mean and variance are roughly what we expect. The mean is around 0, and the variance is around 50.

Problem 2

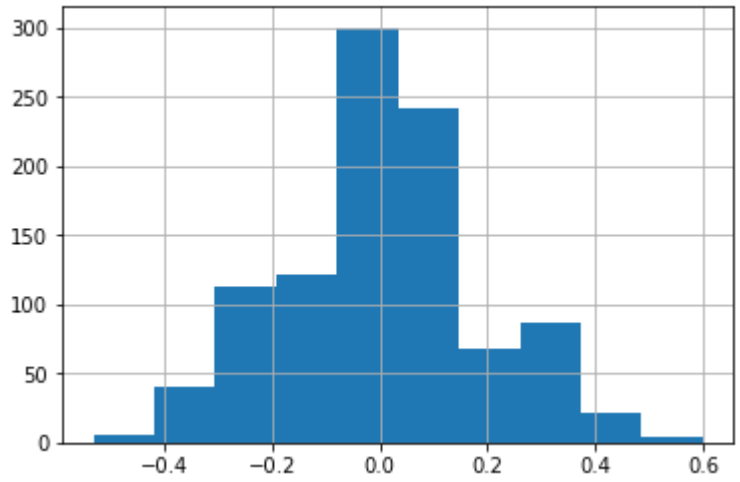
In [88]:
```
n=5
# take n*1000 samples of bernoulli distribution with values {-1,1} and probabi
lity 1/2
dist2 = np.random.choice([1,-1],(n,1000))
# average each set of n samples to sample Zn 1000 times and plot the histogram
bernSeries = pd.Series(dist2.mean(0))
bernSeries.hist()
```

Out[88]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xdf33048>
```



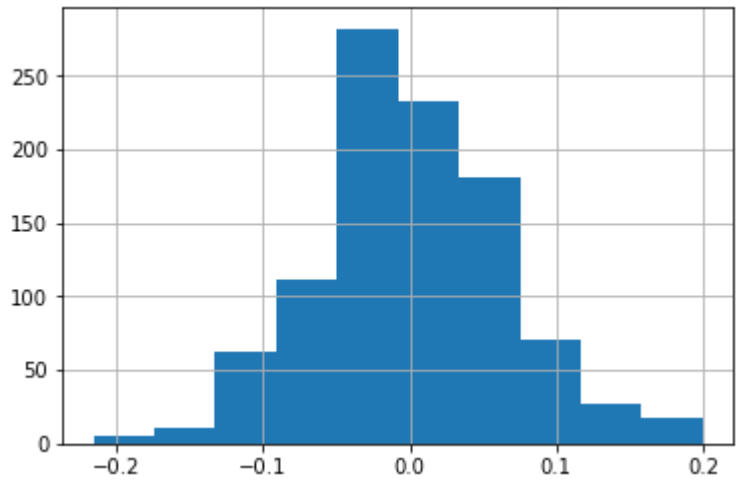
In [89]:
```
# repeat with n = 30
n=30
dist2 = np.random.choice([1,-1],(n,1000))
bernSeries = pd.Series(dist2.mean(0))
bernSeries.hist()
```

Out[89]: <matplotlib.axes._subplots.AxesSubplot at 0xdf33d68>



In [90]:
```
# repeat with n = 250
n=250
dist2 = np.random.choice([1,-1],(n,1000))
bernSeries = pd.Series(dist2.mean(0))
bernSeries.hist()
```

Out[90]: <matplotlib.axes._subplots.AxesSubplot at 0xe55a710>



The distribution looks more gaussian as n becomes larger.

Problem 3

In [91]:
```
# take 25000 samples from gaussian distribution
gauss3 = np.random.normal(0,5,25000)
# determine mean of distribution
mean3 = gauss3.sum()/gauss3.size
mean3
```

Out[91]: 0.069527727372115045

In [92]:
```
# determine standard deviation of distribution by finding the distance of each
 point from the mean
# then averaging the distances and taking the square root
diff3 = gauss3 - mean3
distance3 = diff3**2
stddev3 = np.sqrt(distance3.sum()/distance3.size)
stddev3
```

Out[92]: 4.9605543095702229

The mean and standard deviation are close to the expected values of 0 and 5, respectively.

Problem 4

In [93]:
```
# take 10000 samples from the multivariate normal distribution
multigauss4 = np.random.multivariate_normal([-5,5],[[20,.8],[.8,30]],10000)
multigauss4
```

Out[93]:
```
array([[ -0.36183118,   1.49033198],
       [ -6.88179301,  10.05868906],
       [ -2.49598503,   2.37945389],
       ...,
       [ -8.59782486,   9.48828634],
       [  1.24784861,   0.46857491],
       [ -5.37676679,  -1.62145471]])
```

In [94]:
```
# find the mean of x values and y values separately
mean4 = multigauss4.sum(0)/multigauss4.shape[0]
meanX4 = mean4[0]
meanY4 = mean4[1]
mean4
```

Out[94]: array([-5.10060911, 4.97852824])

In [95]:
```
# find the variance of X by taking the distance of each x point from the mean
 of X and finding the average
# do the same for y
diff4 = multigauss4 - mean4
distance4 = diff4**2
var4 = distance4.sum(0)/distance4.shape[0]
varX4 = var4[0]
varY4 = var4[1]
var4
```

Out[95]:
```
array([ 19.51901015,  30.14726526])
```

In [96]:
```
# find the covariance of X and Y by multiplying (x-E[X]) and (y-E[Y]) for each
 point (x,y) then find the average for all points
prod4 = np.prod(diff4,1)
cov4 = prod4.sum()/prod4.size
cov4
```

Out[96]:
```
1.4215486593721389
```

The mean vector is about (5, 5) and the covariance matrix is about ((20, 0.8),(0.8, 30)), which is what we expect.

Problem 5

In [97]:
```
path='PatientData.csv'
df=pd.read_csv(path,header=None,na_values='?')
```

(a)

In [98]:
```
# number of patients is 452, number of features is 280
df.shape
```

Out[98]:
```
(452, 280)
```

(b)

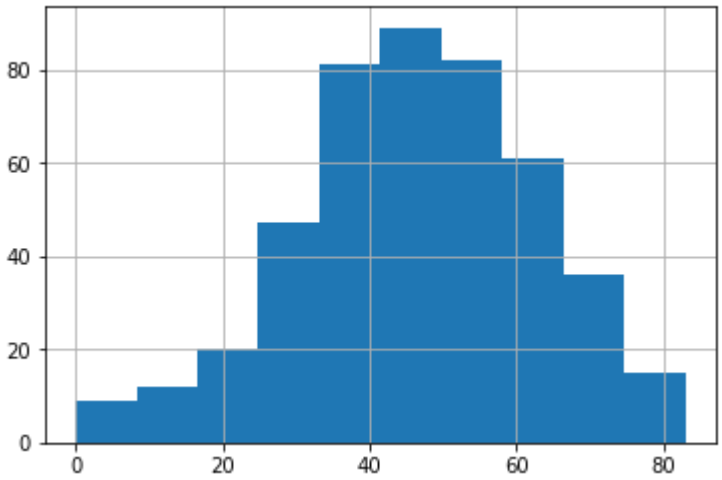
First column

```
In [99]: # show top 8 highest counts of the first column
         df[0].value_counts()[:8]

Out[99]: 46    15
         36    14
         37    14
         47    14
         44    13
         35    13
         45    13
         40    12
         Name: 0, dtype: int64

In [100]: # show histogram of first column/feature
          df[0].hist(bins=10)

Out[100]: <matplotlib.axes._subplots.AxesSubplot at 0xe633080>
```



The first column/feature (df[0]) ranges from 0 to around 80, with the mode being 46. We can assume the first feature to be the age of the patient.

Second column

```
In [101]: # show top highest counts of second column
          df[1].value_counts()

Out[101]: 1    249
          0    203
          Name: 1, dtype: int64
```

Looks like a binary feature but can't make a conclusion on what the feature is with just this information. We will move onto the third column to see if it can help us decipher the second column.

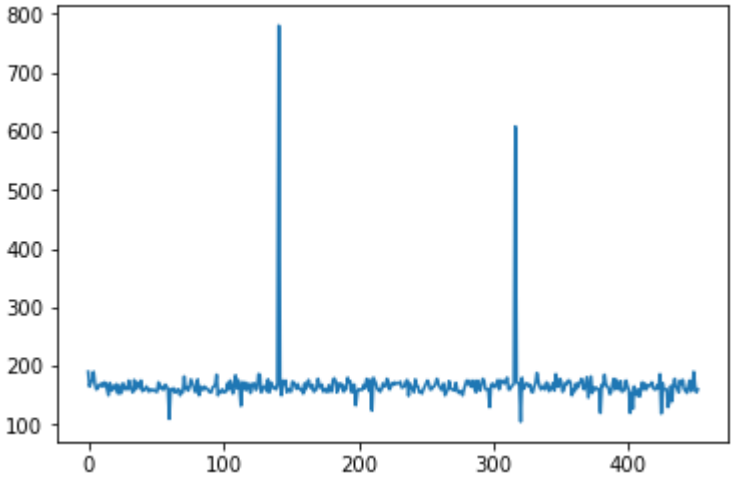
Third column

In [110]:
```
# show top counts for third column
df[2].value_counts()[:10]
```

Out[110]:
```
160    81
165    46
170    40
155    23
175    21
156    19
163    16
162    15
168    15
172    14
Name: 2, dtype: int64
```

In [103]:
```
# plot value vs patient number
plt.plot(df[2])
```

Out[103]:
```
[<matplotlib.lines.Line2D at 0xefbee80>]
```



In [104]:
```
# show the rows where df[2] is big, which would be the two random spikes in th
e data
df[df[2]>200]
```

Out[104]:

	0	1	2	3	4	5	6	7	8	9	...	270	271	272	273	274	275	276	277
141	1	1	780	6	85	165	237	150	106	88	...	0.0	5.0	-4.6	0.0	0.0	1.3	0.7	2.7
316	0	0	608	10	83	126	232	128	60	125	...	-0.7	4.5	-5.5	0.0	0.0	0.5	2.5	-11.8

2 rows × 280 columns

In [105]:
```
df[2].mean()
```

Out[105]:
```
166.18805309734512
```

Besides the two outlier patients which seem to be errors, most patients seem to be around the 155-175 range for column three. We can conclude that this feature is most likely the height in centimeters of the patient.

Back to second column

In [106]:
```
# show the averages of each column, grouped by the value of column one (the bi
nary feature)
df2 = df.groupby(1).mean()
df2
```

Out[106]:

	0	2	3	4	5	6	7	8
1								
0	47.546798	171.315271	72.724138	94.650246	157.472906	364.546798	177.231527	92.3
1	45.594378	162.008032	64.457831	84.248996	153.261044	369.377510	164.012048	88.1

2 rows × 279 columns

We can see that the average height of patients with df[1]=0 is 171cm and patients with df[1]=1 is 162 cm. Thus, we can assume that the second feature is the gender of the patient. Specifically, the value 0 indicates male and value 1 indicates female.

Fourth column

Looking online at national averages for men and women, we can also conclude that the fourth column df[3] is the feature weight in kilograms.

(c)

In [107]:
```
# fill the missing values with the averages of their respective columns
filledDF = df.fillna(df.mean())
filledDF
```

Out[107]:

	0	1	2	3	4	5	6	7	8	9	...	270	271	272	273	274	275	276
0	75	0	190	80	91	193	371	174	121	-16	...	0.0	9.0	-0.9	0.0	0.0	0.9	2.9
1	56	1	165	64	81	174	401	149	39	25	...	0.0	8.5	0.0	0.0	0.0	0.2	2.1
2	54	0	172	95	138	163	386	185	102	96	...	0.0	9.5	-2.4	0.0	0.0	0.3	3.4
3	55	0	175	94	100	202	380	179	143	28	...	0.0	12.2	-2.2	0.0	0.0	0.4	2.6
4	75	0	190	80	88	181	360	177	103	-16	...	0.0	13.1	-3.6	0.0	0.0	-0.1	3.9
5	13	0	169	51	100	167	321	174	91	107	...	-0.6	12.2	-2.8	0.0	0.0	0.9	2.2
6	40	1	160	52	77	129	377	133	77	77	...	0.0	6.5	0.0	0.0	0.0	0.4	1.0
7	49	1	162	54	78	0	376	157	70	67	...	0.0	8.2	-1.9	0.0	0.0	0.1	0.5
8	44	0	168	56	84	118	354	160	63	61	...	0.0	7.0	-1.3	0.0	0.0	0.6	2.1
9	50	1	167	67	89	130	383	156	73	85	...	-0.6	10.8	-1.7	0.0	0.0	0.8	0.9
10	62	0	170	72	102	135	401	156	83	72	...	-0.5	9.0	-2.0	0.0	0.0	0.8	0.9
11	45	1	165	86	77	143	373	150	65	12	...	0.0	4.4	-2.2	0.0	0.0	0.5	1.5
12	54	1	172	58	78	155	382	163	81	-24	...	0.0	6.3	-2.1	0.0	0.0	0.8	0.5
13	30	0	170	73	91	180	355	157	104	68	...	-0.9	12.3	0.0	0.0	0.0	0.4	2.1
14	44	1	160	88	77	158	399	163	94	46	...	-0.6	12.4	0.0	0.0	0.0	0.3	1.7
15	47	1	150	48	75	132	350	169	65	36	...	0.0	7.7	-0.8	0.0	0.0	0.6	1.7
16	47	0	171	59	82	145	347	169	61	77	...	0.0	9.4	-1.7	0.0	0.0	0.6	2.3
17	46	1	158	58	70	120	353	122	52	57	...	0.0	6.6	0.0	0.0	0.0	0.3	0.7
18	73	0	165	63	91	154	392	175	83	73	...	0.0	5.7	0.0	0.0	0.0	0.4	0.5
19	57	1	166	72	82	181	399	158	79	-12	...	0.0	7.7	-0.9	0.0	0.0	0.5	1.8
20	28	1	160	58	83	251	383	189	183	50	...	-0.6	9.1	-1.4	0.0	0.0	0.6	3.3
21	45	0	169	67	90	122	336	177	78	81	...	-0.6	8.3	-1.8	0.0	0.0	0.8	1.1
22	36	1	153	75	71	132	364	169	82	62	...	0.0	8.9	-1.0	0.0	0.0	0.5	1.7
23	57	1	165	59	75	157	406	143	92	4	...	0.0	6.7	-0.5	0.0	0.0	0.4	1.1
24	40	1	153	55	82	140	388	149	82	52	...	0.0	13.6	0.0	0.0	0.0	0.5	2.5
25	44	0	169	80	109	128	382	195	60	-34	...	0.0	6.9	0.0	0.0	0.0	0.4	1.3
26	34	0	170	73	94	186	373	224	125	90	...	0.0	15.3	-1.1	0.0	0.0	0.6	2.6
27	31	1	160	54	95	161	407	168	83	10	...	0.0	12.7	-1.8	0.0	0.0	0.3	3.2
28	56	1	164	65	90	164	420	381	99	-8	...	0.0	5.4	0.0	0.0	0.0	0.4	-1.4
29	51	1	160	83	96	147	400	301	82	-37	...	0.0	7.3	-3.9	0.0	0.0	0.5	-1.1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
422	29	1	162	57	83	164	359	154	69	64	...	0.0	14.1	-2.2	0.0	0.0	0.5	3.0

	0	1	2	3	4	5	6	7	8	9	...	270	271	272	273	274	275	276
423	51	0	186	95	94	203	367	171	106	-7	...	0.0	9.6	-3.5	0.0	0.0	1.0	1.6
424	7	0	119	21	140	157	438	226	81	-40	...	0.0	10.0	-2.1	0.0	0.0	1.0	5.5
425	36	0	171	93	87	150	362	177	96	44	...	0.0	10.3	-0.8	0.0	0.0	0.6	3.0
426	35	1	160	53	55	163	340	162	102	40	...	0.0	8.7	-0.5	0.0	0.0	0.5	2.3
427	58	0	160	65	133	148	417	260	92	-158	...	-0.4	6.4	-3.5	0.0	0.0	0.4	0.8
428	64	0	160	63	83	0	364	120	90	29	...	0.0	6.7	-0.4	0.0	0.0	0.3	0.4
429	8	1	130	24	77	125	358	159	70	87	...	0.0	11.3	-2.1	0.0	0.0	0.7	3.6
430	11	0	138	29	123	145	361	221	80	112	...	-3.4	19.6	-4.2	0.0	0.0	0.2	1.8
431	47	0	166	56	79	145	381	173	101	52	...	0.0	8.5	0.0	0.0	0.0	0.6	1.2
432	11	0	140	42	88	123	362	228	81	-18	...	0.0	17.1	-7.1	0.0	0.0	0.7	5.5
433	70	0	167	60	80	149	290	128	93	-67	...	0.0	2.7	-5.4	0.0	0.0	0.3	-0.2
434	20	0	178	65	88	155	360	163	71	-22	...	-0.5	10.2	0.0	0.0	0.0	0.5	0.4
435	39	1	164	62	79	155	367	153	95	50	...	0.0	9.7	-0.7	0.0	0.0	0.8	1.3
436	32	1	164	57	77	144	340	148	82	27	...	-0.6	9.9	-0.6	0.0	0.0	0.5	2.4
437	35	1	155	63	87	142	391	137	88	66	...	0.0	10.7	0.0	0.0	0.0	1.0	2.1
438	37	0	175	82	88	146	357	179	72	1	...	-0.4	13.5	-1.2	0.0	0.0	0.5	0.6
439	49	1	168	66	94	170	383	152	115	92	...	0.0	8.2	-0.7	0.0	0.0	0.8	1.7
440	37	0	176	72	88	153	389	172	89	67	...	-0.9	16.6	-3.4	0.0	0.0	0.7	1.8
441	37	1	160	50	74	143	374	146	75	68	...	0.0	11.4	-0.9	0.0	0.0	0.7	1.8
442	65	1	160	50	85	143	363	146	84	-40	...	0.0	6.6	-6.1	0.0	0.0	0.5	0.5
443	41	1	154	75	88	157	384	132	112	65	...	-0.4	10.5	-2.5	0.0	0.0	0.5	1.4
444	29	0	166	63	81	143	325	218	74	24	...	0.0	7.8	-1.3	0.0	0.0	0.5	2.3
445	45	0	175	75	91	134	376	160	83	91	...	0.0	7.1	-2.4	0.0	0.0	-0.4	1.3
446	20	1	157	57	81	151	363	166	80	43	...	0.0	7.2	-0.7	0.0	0.0	0.5	2.3
447	53	1	160	70	80	199	382	154	117	-37	...	0.0	4.3	-5.0	0.0	0.0	0.7	0.6
448	37	0	190	85	100	137	361	201	73	86	...	0.0	15.6	-1.6	0.0	0.0	0.4	2.4
449	36	0	166	68	108	176	365	194	116	-85	...	0.0	16.3	-28.6	0.0	0.0	1.5	1.0
450	32	1	155	55	93	106	386	218	63	54	...	-0.4	12.0	-0.7	0.0	0.0	0.5	2.4
451	78	1	160	70	79	127	364	138	78	28	...	0.0	10.4	-1.8	0.0	0.0	0.5	1.6

452 rows × 280 columns

(d)

You can find which features strongly influence the patient condition by finding the correlation matrix and seeing which columns have a high correlation with the last column

In [108]:
```
correlation = filledDF.corr()
correlation[279].sort_values(ascending=False)[:4]
```

Out[108]:
```
279    1.000000
90     0.368876
4      0.323879
92     0.313982
Name: 279, dtype: float64
```

The top 3 features that influence the condition are features 90, 4, and 92.

Written Questions

① a) $\frac{1}{4}+\frac{1}{3}=\frac{7}{12}$

b) $P(A|B)=\frac{P(A\cap B)}{P(B)}$ $P(X=1|Y=1)=\frac{P(X=1\cap Y=1)}{P(Y=1)}=\frac{1/3}{1/6+1/3}=\frac{1/3}{1/2}=\boxed{\frac{2}{3}}$

c) $Var(X)=E[X^2]-E^2[X]$

$E[X^2]=0^2\cdot\frac{1}{4}+0^2\cdot\frac{1}{6}+1^2\cdot\frac{1}{4}+1^2\cdot\frac{1}{3}=\frac{7}{12}$

$E^2[X]=\left(0\cdot\frac{1}{4}+0\cdot\frac{1}{6}+1\cdot\frac{1}{4}+1\cdot\frac{1}{3}\right)^2=\left(\frac{7}{12}\right)^2=\frac{49}{144}$

$\frac{7}{12}-\frac{49}{144}=\boxed{\frac{35}{144}}$

d) $Var(X|Y=1)=E[(X|Y=1)^2]-E^2[(X|Y=1)]$

$E[(X|Y=1)^2]=0^2\cdot\frac{1/6}{1/2}+1^2\cdot\frac{1/3}{1/2}=\frac{2}{3}$

$E^2[(X|Y=1)]=\left(\frac{2}{3}\right)^2=\frac{4}{9}$

$\frac{2}{3}-\frac{4}{9}=\boxed{\frac{2}{9}}$

e) $E[X^3+X^2+3Y^7|Y=1]=E[X^3|Y=1]+E[X^2|Y=1]+3E[Y^7|Y=1]$

$E[X^3|Y=1]=1^3\cdot\frac{1/3}{1/2}=\frac{2}{3}$

$E[X^2|Y=1]=1^2\cdot\frac{1/3}{1/2}=\frac{2}{3}$

$3E[Y^7|Y=1]=3$

$\frac{2}{3}+\frac{2}{3}+3=\boxed{\frac{13}{3}}$

② $N=\vec{V_1}\times\vec{V_2}$

$$\begin{vmatrix} i & j & k \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{vmatrix}=0i+1j+-1k=0i+1j-1k=[0,1,-1]$$

$Proj\,P_1=P_1-Proj_N P_1=[3,3,3]-\frac{P_1\cdot N}{\|N\|=\vec{N}}=[3,3,3]$

$Proj\,P_2=[1,2,3]-\frac{-1}{2}[0,1,-1]=[1,2.5,2.5]$

$Proj\,P_3=[0,0,1]-\frac{-1}{2}[0,1,-1]=[0,0.5,0.5]$

③ $P(X=1)=2/3$ $P(X=0)=1/3$

$P(x\leq 50)=\sum_{i=0}^{50}\binom{100}{i}\left(\frac{2}{3}\right)^i\left(\frac{1}{3}\right)^{100-i}=0.000419$