EE460J Lab 1

James Lu, Soroush Famili, Nithanth Ram

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

1. Create 1000 samples from a Gaussian distribution with mean -10 and standard deviation 5. Create another 1000 samples from another independent Gaussian with mean 10 and standard deviation 5.

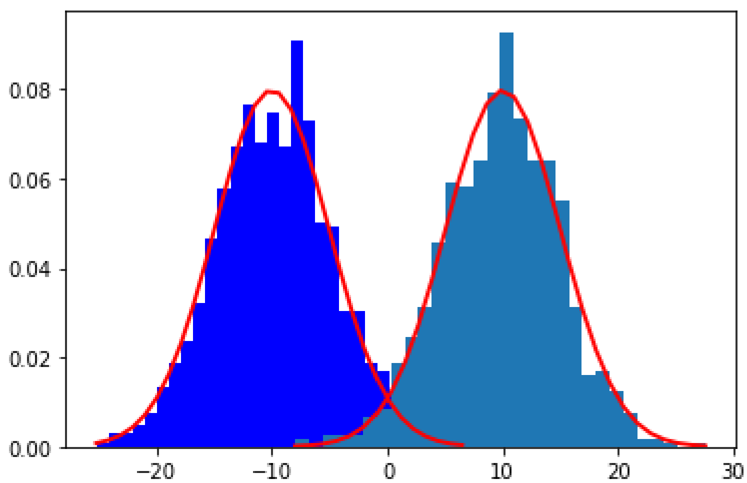
In [3]:
```
#generate random samples
n = 1000

mean_1 = -10
std_1 = 5
samples_1 = np.random.normal(mean_1, std_1, n)

mean_2 = 10
std_2 = 5
samples_2 = np.random.normal(mean_2, std_2, n)

#plot values
count_1, bins_1, ignored_1 = plt.hist(samples_1, bins=30, density=True, color=
"blue")
plt.plot(bins_1, 1/(std_1 * np.sqrt(2 * np.pi)) *
            np.exp( - (bins_1 - mean_1)**2 / (2 * std_1**2) ),
            linewidth=2, color='r')
count_2, bins_2, ignored_2 = plt.hist(samples_2, bins=30, density=True)
plt.plot(bins_2, 1/(std_2 * np.sqrt(2 * np.pi)) *
            np.exp( - (bins_2 - mean_2)**2 / (2 * std_2**2) ),
            linewidth=2, color='r')

plt.show()
```

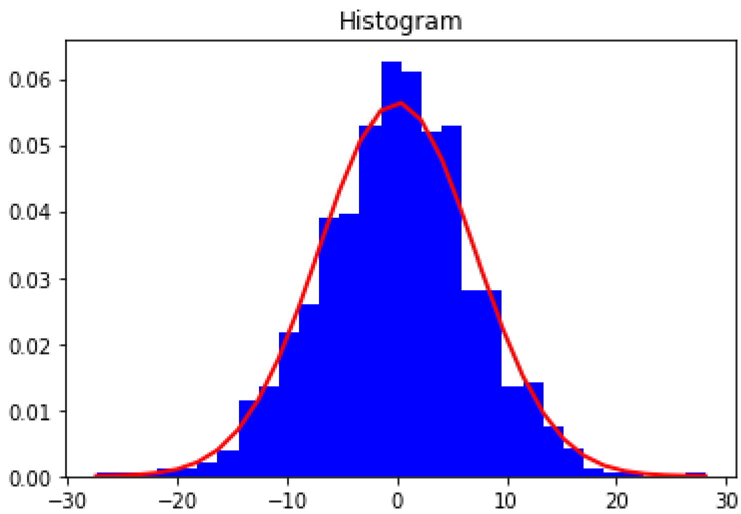


(a) Take the sum of 2 these Gaussians by adding the two sets of 1000 points, point by point, and plot the histogram of the resulting 1000 points. What do you observe?

In [4]:
```
#compute sum
samples = samples_1 + samples_2
mean = 0
std = np.sqrt(50)

#plot values
count, bins, ignored = plt.hist(samples, bins=30, density=True, color="blue")
plt.plot(bins, 1/(std * np.sqrt(2 * np.pi)) *
            np.exp( - (bins - mean)**2 / (2 * std**2) ),
            linewidth=2, color='r')

plt.title("Histogram")
plt.show()
```



I realized that the sum of these samples, which were taken from normal distributions, resulted in what looks like samples from a new normal distribution.

(b) Estimate the mean and the variance of the sum.

Because both these samples were taken from normal distributions, the new mean μ is the sum of the previous means ($\mu = \mu_1 + \mu_2 \approx 0$), and the new variance σ^2 is the sum of the previous variances ($\sigma^2 = \sigma_1^2 + \sigma_2^2 \approx 50$), meaning that the new standard deviation $\sigma \approx 7.07$

In [5]:
```
print("Validating findings:")
print("μ = ", samples.mean())
print("σ^2 = ", samples.var())
print("σ = ", np.sqrt(samples.var()))
```

```
Validating findings:
μ =  0.04448055705149889
σ^2 =  48.46161965397354
σ =  6.961438044971279
```

2. Central Limit Theorem - Let X_i be an i.i.d. Bernoulli random variable with value {-1,1}. Look at the random variable Z_n. By taking 1000 draws from Zn, plot its histogram. Check that for small n (say, 5-10) Zn does not look that much like a Gaussian, but when n is bigger (already by the time n = 30 or 50) it looks much more like a Gaussian. Check also for much bigger n: n = 250, to see that at this point, one can really see the bell curve.

In [6]:

```
# function for generating samples from Z_n
def Z_n(samples, n):
    bernoulli = [-1,1]  #values for bernoulli rv
    result = []
    for i in range(samples):
        sample = 0 #generate sample of Z_n
        for j in range(n):
            sample += np.random.choice(bernoulli)  #sum of bernoulli rvs
        sample /= np.sqrt(n) #normalize
        result.append(sample)

    return result
```

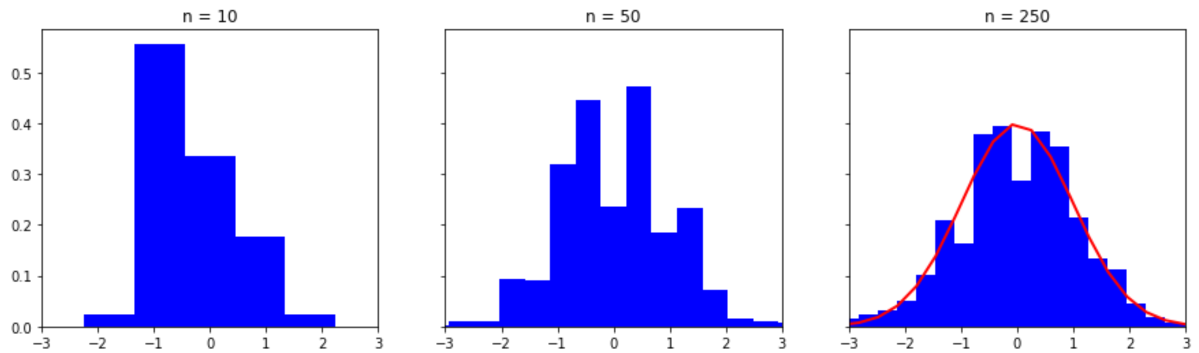
In [7]:
```
samples = 1000
fig, ax = plt.subplots(1,3, sharex=True, sharey=True)  # three plots
fig.set_figwidth(15)

n = 5  #num. of bernoulli rvs
result = Z_n(samples, n)
count, bins, ignored = ax[0].hist(result, bins=5, density=True, color="blue")
ax[0].title.set_text("n = 10")
ax[0].set_xlim(-3,3)

n = 50
result = Z_n(samples, n)
count, bins, ignored = ax[1].hist(result, bins=15, density=True, color="blue")
ax[1].title.set_text("n = 50")

n = 250
result = Z_n(samples, n)
count, bins, ignored = ax[2].hist(result, bins=20, density=True, color="blue")
ax[2].plot(bins, 1/(1 * np.sqrt(2 * np.pi)) *
            np.exp( - (bins - 0)**2 / (2 * 1**2) ),
            linewidth=2, color='r')
ax[2].title.set_text("n = 250")

plt.show()
```



3. Estimate the mean and standard deviation from 1 dimensional data: generate 25,000 samples from a Gaussian distribution with mean 0 and standard deviation 5. Then estimate the mean and standard deviation of this gaussian using elementary numpy commands, i.e., addition, multiplication, division (do not use a command that takes data and returns the mean or standard deviation).

In [8]:
```
#generate random samples
n = 25000
mean = 0
std = 5
samples = np.random.normal(mean, std, n)
```

In [9]:
```
mean_measured = 0  #compute mean for samples
for s in samples:
    mean_measured += s / n

var_measured = 0  #compute variance: squared differences from mean
for s in samples:
    var_measured += (s - mean_measured)**2 / n

std_measured = np.sqrt(var_measured)  #compute standard deviation
```

In [10]:
```
print(mean_measured)
print(std_measured)
```

```
-0.012868351062482098
5.002045257300916
```

4. Estimate the mean and covariance matrix for multi-dimensional data: generate 10,000 samples of 2 dimensional data from the Gaussian distribution

In [11]:
```
n = 10000
mean = [-5,5]
covariance = [[20, .8],[.8, 30]]
samples = np.random.multivariate_normal(mean, covariance, n).T
```

In [12]:
```
mean_measured = [0, 0]
for s in samples[0]:
    mean_measured[0] += s / n  # compute mean for X

for s in samples[1]:
    mean_measured[1] += s / n  # compute mean for y
```

In [13]:
```
print("X mean: " + str(mean_measured[0]))
print("Y mean: " + str(mean_measured[1]))
```

```
X mean: -4.994312728139394
Y mean: 4.959764232527516
```

In [14]:
```
covariance_measured = [[0,0],[0,0]]
for s in samples[0]:
    covariance_measured[0][0] += (s - mean_measured[0])**2 / n  # compute vari
ance for X

for s in samples[1]:
    covariance_measured[1][1] += (s - mean_measured[1])**2 / n  # compute vari
ance for y

for i in range(n):
    covariance_measured[0][1] += (samples[0][i] - mean_measured[0])*(samples[1
][i] - mean_measured[0]) / n  # compute covariance
    covariance_measured[1][0] = covariance_measured[0][1]
```

In [15]:
```
print(np.array(covariance_measured))
```

```
[[20.28519291  0.67334107]
 [ 0.67334107 29.8209368 ]]
```

5. Each row is a patient and the last column is the condition that the patient has. Do data exploration using Pandas and other visualization tools to understand what you can about the dataset.

In [16]:
```
# make sure to add header=None, otherwise the first row is treated as a header
patients = pd.read_csv("PatientData.csv", header=None)
print(patients.shape) # 452 patients, 279 features
display(patients)
```

```
(452, 280)
```

	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	0.9	2.9
1	56	1	165	64	81	174	401	149	39	25	...	0.0	8.5	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	0.3	3.4
3	55	0	175	94	100	202	380	179	143	28	...	0.0	12.2	-2.2	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	-0.1	3.9
5	13	0	169	51	100	167	321	174	91	107	...	-0.6	12.2	-2.8	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	0.4	1.0
7	49	1	162	54	78	0	376	157	70	67	...	0.0	8.2	-1.9	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	0.6	2.1
9	50	1	167	67	89	130	383	156	73	85	...	-0.6	10.8	-1.7	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	0.8	0.9
11	45	1	165	86	77	143	373	150	65	12	...	0.0	4.4	-2.2	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	0.8	0.5
13	30	0	170	73	91	180	355	157	104	68	...	-0.9	12.3	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	0.3	1.7
15	47	1	150	48	75	132	350	169	65	36	...	0.0	7.7	-0.8	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	0.6	2.3
17	46	1	158	58	70	120	353	122	52	57	...	0.0	6.6	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	0.4	0.5
19	57	1	166	72	82	181	399	158	79	-12	...	0.0	7.7	-0.9	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	0.6	3.3
21	45	0	169	67	90	122	336	177	78	81	...	-0.6	8.3	-1.8	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	0.5	1.7
23	57	1	165	59	75	157	406	143	92	4	...	0.0	6.7	-0.5	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	0.5	2.5
25	44	0	169	80	109	128	382	195	60	-34	...	0.0	6.9	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	0.6	2.6
27	31	1	160	54	95	161	407	168	83	10	...	0.0	12.7	-1.8	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	0.4	-1.4
29	51	1	160	83	96	147	400	301	82	-37	...	0.0	7.3	-3.9	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	0.5	3.0
423	51	0	186	95	94	203	367	171	106	-7	...	0.0	9.6	-3.5	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	1.0	5.5
425	36	0	171	93	87	150	362	177	96	44	...	0.0	10.3	-0.8	0.0	0	0.6	3.0

	0	1	2	3	4	5	6	7	8	9	...	270	271	272	273	274	275	276
426	35	1	160	53	55	163	340	162	102	40	...	0.0	8.7	-0.5	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	0.4	0.8
428	64	0	160	63	83	0	364	120	90	29	...	0.0	6.7	-0.4	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	0.7	3.6
430	11	0	138	29	123	145	361	221	80	112	...	-3.4	19.6	-4.2	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	0.6	1.2
432	11	0	140	42	88	123	362	228	81	-18	...	0.0	17.1	-7.1	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	0.3	-0.2
434	20	0	178	65	88	155	360	163	71	-22	...	-0.5	10.2	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	0.8	1.3
436	32	1	164	57	77	144	340	148	82	27	...	-0.6	9.9	-0.6	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	1.0	2.1
438	37	0	175	82	88	146	357	179	72	1	...	-0.4	13.5	-1.2	0.0	0	0.5	0.6
439	49	1	168	66	94	170	383	152	115	92	...	0.0	8.2	-0.7	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	0.7	1.8
441	37	1	160	50	74	143	374	146	75	68	...	0.0	11.4	-0.9	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	0.5	0.5
443	41	1	154	75	88	157	384	132	112	65	...	-0.4	10.5	-2.5	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	0.5	2.3
445	45	0	175	75	91	134	376	160	83	91	...	0.0	7.1	-2.4	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	0.5	2.3
447	53	1	160	70	80	199	382	154	117	-37	...	0.0	4.3	-5.0	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	0.4	2.4
449	36	0	166	68	108	176	365	194	116	-85	...	0.0	16.3	-28.6	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	0.5	2.4
451	78	1	160	70	79	127	364	138	78	28	...	0.0	10.4	-1.8	0.0	0	0.5	1.6

452 rows × 280 columns

(a) How many patients and how many features are there?

452 patients, 279 features

(b) What is the meaning of the first 4 features? See if you can understand what they mean.

First feature could be age of patient. Second could be male or female. Third could be height in centimeters. Fourth could be mass in kilograms.

(c) Are there missing values? Replace them with the average of the corresponding feature column.

There are missing values.

In [17]:
```
print("Missing values: ",'?' in patients.values)
patients.replace('?', np.nan, inplace=True)  # replace question marks in dataframe with NaNs
patients = patients.astype(float)
patients.fillna(patients.mean(), inplace=True)  # replace NaNs with the average of column
```

```
Missing values:  True
```

(d) How could you test which features strongly influence the patient condition and which do not?

The features are strongly correlated or inversely correlated with the patient's condition have the strongest influence. The features that have little to no correlation do not have a strong influence.

In [18]:
```
patients_corr = patients.corr()  # compute correlation matrix between all feat
ures
display(patients_corr.round(decimals=2))
```

	0	1	2	3	4	5	6	7	8	9	...	270	271	272	27
0	1.00	-0.06	-0.11	0.38	-0.00	0.04	0.20	0.03	0.10	-0.27	...	0.16	-0.16	0.08	0.0
1	-0.06	1.00	-0.12	-0.25	-0.34	-0.05	0.07	-0.18	-0.08	0.07	...	0.23	-0.04	0.09	0.0
2	-0.11	-0.12	1.00	-0.07	-0.01	0.01	-0.24	-0.04	0.03	0.06	...	-0.02	-0.07	-0.09	-0.0
3	0.38	-0.25	-0.07	1.00	0.10	0.12	0.12	0.15	0.12	-0.17	...	0.05	-0.03	0.05	0.0
4	-0.00	-0.34	-0.01	0.10	1.00	0.02	0.22	0.40	0.05	-0.15	...	-0.20	0.09	-0.23	-0.0
5	0.04	-0.05	0.01	0.12	0.02	1.00	0.08	0.07	0.67	-0.01	...	-0.01	0.00	-0.07	0.0
6	0.20	0.07	-0.24	0.12	0.22	0.08	1.00	0.17	0.06	-0.03	...	0.04	0.12	0.12	-0.0
7	0.03	-0.18	-0.04	0.15	0.40	0.07	0.17	1.00	0.06	-0.10	...	-0.07	0.07	-0.05	0.0
8	0.10	-0.08	0.03	0.12	0.05	0.67	0.06	0.06	1.00	-0.06	...	-0.01	0.01	-0.09	0.1
9	-0.27	0.07	0.06	-0.17	-0.15	-0.01	-0.03	-0.10	-0.06	1.00	...	-0.18	0.29	0.30	-0.0
10	0.01	-0.14	-0.00	-0.04	0.03	0.12	0.16	0.04	0.15	-0.05	...	-0.03	-0.06	-0.06	0.0
11	-0.04	0.01	-0.11	-0.03	0.04	-0.03	-0.10	-0.11	0.03	0.05	...	0.00	-0.07	0.07	-0.0
12	-0.28	0.03	0.02	-0.21	-0.08	0.02	-0.09	-0.11	-0.07	0.71	...	-0.22	0.17	0.13	-0.0
13	-0.03	-0.07	0.01	0.06	0.08	0.10	-0.07	0.08	0.08	-0.18	...	0.08	-0.16	-0.22	-0.0
14	-0.19	0.06	0.29	-0.17	-0.01	-0.04	-0.65	0.02	0.04	0.01	...	-0.05	-0.12	-0.24	0.0
15	-0.02	-0.13	-0.00	0.04	0.12	0.02	0.02	0.02	0.07	-0.06	...	-0.34	-0.12	-0.03	-0.0
16	0.20	0.02	-0.09	0.12	0.31	-0.07	0.30	0.25	-0.03	-0.27	...	0.20	0.13	0.20	0.0
17	-0.12	-0.15	0.04	-0.02	0.19	0.05	-0.08	-0.04	0.04	0.21	...	-0.11	0.03	-0.30	-0.0
18	0.03	-0.02	0.01	0.12	0.05	0.01	0.01	0.01	-0.01	0.06	...	-0.04	0.03	0.04	-0.0
19	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	Na
20	0.15	-0.09	-0.08	0.15	0.42	-0.03	0.23	0.29	0.01	-0.26	...	-0.05	0.13	0.04	-0.0
21	0.04	0.04	0.00	0.12	0.02	-0.16	-0.10	0.02	-0.16	0.01	...	0.02	0.08	0.03	-0.0
22	0.13	0.01	-0.01	-0.00	0.20	0.05	0.08	0.07	0.08	0.01	...	0.05	-0.02	0.02	0.3
23	0.01	-0.07	0.01	0.00	0.00	0.01	0.08	-0.02	-0.07	-0.07	...	-0.04	-0.08	0.00	-0.0
24	0.09	-0.01	-0.01	-0.01	0.02	-0.11	0.01	-0.03	0.05	-0.06	...	0.03	-0.06	0.03	-0.0
25	0.06	-0.07	0.00	0.01	0.17	0.01	0.09	0.13	-0.01	-0.06	...	0.03	0.02	-0.02	-0.0
26	0.03	-0.10	0.01	0.07	0.24	0.00	0.08	0.20	0.01	0.05	...	-0.02	0.02	0.04	-0.0
27	-0.04	-0.14	-0.01	0.09	0.09	0.00	-0.06	0.01	-0.00	0.13	...	-0.50	0.07	0.10	0.0
28	0.21	0.02	-0.06	0.13	0.26	-0.00	0.25	0.15	-0.04	0.31	...	0.07	0.23	0.30	-0.0
29	-0.13	-0.12	-0.01	-0.06	0.29	-0.03	0.06	0.16	0.09	-0.43	...	0.06	-0.14	-0.40	-0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
250	-0.00	0.16	-0.02	-0.09	-0.05	-0.01	-0.01	-0.16	-0.00	-0.09	...	0.26	0.14	-0.07	0.0
251	-0.17	-0.23	0.12	-0.11	0.04	-0.03	-0.01	-0.05	-0.00	0.20	...	-0.27	0.51	0.06	-0.0
252	0.02	0.27	-0.06	0.05	-0.38	0.06	-0.02	-0.15	-0.01	0.09	...	0.08	-0.06	0.47	-0.0
253	-0.19	-0.04	-0.03	-0.11	-0.05	-0.05	-0.08	-0.05	-0.10	0.15	...	-0.05	0.06	0.05	0.0

	0	1	2	3	4	5	6	7	8	9	...	270	271	272	27
254	-0.01	0.03	-0.00	0.02	0.01	-0.02	-0.06	0.01	-0.00	-0.03	...	-0.03	-0.01	0.00	0.0
255	-0.05	-0.04	0.05	-0.07	-0.06	0.04	0.01	-0.00	0.15	0.10	...	-0.15	0.01	0.03	-0.0
256	-0.15	-0.13	0.05	-0.05	0.09	0.01	0.08	0.08	-0.05	0.07	...	-0.08	0.12	0.15	0.0
257	-0.15	0.08	0.01	-0.05	-0.36	0.01	-0.07	-0.22	-0.02	0.22	...	-0.05	0.39	0.36	-0.0
258	-0.22	-0.04	0.02	-0.06	-0.14	0.01	0.04	-0.03	-0.06	0.20	...	-0.09	0.37	0.37	-0.0
259	-0.22	-0.05	0.12	-0.10	-0.02	0.12	-0.09	-0.02	0.06	-0.09	...	-0.03	-0.39	-0.20	0.0
260	0.06	0.19	-0.04	-0.01	-0.13	-0.05	0.03	-0.08	-0.03	-0.04	...	0.75	0.03	-0.01	0.0
261	-0.18	-0.20	0.05	-0.08	0.07	-0.00	0.07	0.02	-0.02	0.25	...	-0.27	0.88	-0.01	-0.0
262	0.03	0.22	-0.10	0.01	-0.28	-0.04	0.06	-0.12	-0.07	0.23	...	-0.08	-0.05	0.81	-0.0
263	0.08	0.05	-0.01	-0.02	0.21	0.00	0.09	0.10	0.02	0.02	...	0.02	0.03	0.03	0.0
264	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	Na
265	-0.05	0.00	0.04	-0.06	-0.08	0.08	-0.02	-0.00	0.21	0.08	...	-0.12	-0.01	-0.09	-0.0
266	-0.26	-0.03	0.01	-0.13	-0.06	0.07	0.02	-0.04	-0.01	0.09	...	-0.02	0.10	0.03	-0.0
267	-0.07	0.02	-0.03	-0.02	-0.08	-0.03	0.13	-0.03	-0.05	0.31	...	-0.01	0.67	0.47	-0.0
268	-0.23	-0.02	-0.04	-0.09	-0.06	0.01	0.13	0.01	-0.05	0.26	...	-0.02	0.51	0.34	-0.0
269	-0.22	-0.01	0.13	-0.12	-0.26	0.11	-0.19	-0.18	0.05	-0.09	...	-0.01	-0.38	-0.26	0.0
270	0.16	0.23	-0.02	0.05	-0.20	-0.01	0.04	-0.07	-0.01	-0.18	...	1.00	-0.19	-0.08	0.0
271	-0.16	-0.04	-0.07	-0.03	0.09	0.00	0.12	0.07	0.01	0.29	...	-0.19	1.00	0.02	-0.0
272	0.08	0.09	-0.09	0.05	-0.23	-0.07	0.12	-0.05	-0.09	0.30	...	-0.08	0.02	1.00	-0.0
273	0.09	0.03	-0.00	0.05	-0.01	0.07	-0.00	0.01	0.10	-0.07	...	0.04	-0.09	-0.01	1.0
274	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	Na
275	-0.04	0.01	0.07	-0.05	-0.07	0.14	-0.03	0.05	0.25	0.08	...	-0.09	-0.01	-0.09	-0.0
276	-0.27	0.07	-0.01	-0.14	-0.22	0.06	-0.04	-0.18	0.02	0.06	...	0.12	0.09	-0.07	-0.0
277	0.02	0.03	-0.09	0.06	0.13	-0.03	0.26	0.13	-0.02	0.30	...	0.00	0.67	0.56	-0.0
278	-0.20	0.05	-0.09	-0.05	-0.08	0.02	0.15	-0.01	0.00	0.26	...	0.08	0.56	0.34	-0.0
279	-0.09	-0.18	0.01	-0.09	0.32	-0.10	0.03	0.10	-0.12	0.02	...	-0.16	0.04	-0.07	-0.0

280 rows × 280 columns

In [19]:
```
features_corr = patients_corr[patients_corr.shape[1] - 1]  # correlation with
 label (last column)
strongly_corr_features = []
for i in range(features_corr.shape[0]):
    if features_corr[i] >= 0.7 or features_corr[i] <= -0.7:  # define highly c
orrelated as ccoef>=0.7
        strongly_corr_features.append(i)  # find features that have a high cor
relation with the label
```

In [20]:
```
strongly_corr_features
```

Out[20]:
```
[279]
```

In [ ]:

Lab 1

Written Questions

1.

	X=0	X=1
Y=0	$\frac{1}{4}$	$\frac{1}{4}$
Y=1	$\frac{1}{6}$	$\frac{1}{3}$

(a)

$$P(X=1) = \frac{1}{4} + \frac{1}{3} \boxed{= \frac{7}{12}}$$

(b)

$$P(X=1 \mid Y=1) = \frac{\frac{1}{3}}{\frac{1}{6}+\frac{1}{3}} = \frac{2}{1+2} \boxed{= \frac{2}{3}}$$

(c)

$$Var[X] = E[X^2] - E[X]^2$$

$$= \sum_x \sum_y x^2 \cdot f(x,y) - \left[\sum_x \sum_y x \cdot f(x,y)\right]^2$$

$$= \left[(1)^2\left(\frac{1}{4}\right) + (1)^2\left(\frac{1}{3}\right)\right] - \left[(1)\left(\frac{1}{4}\right) + (1)\left(\frac{1}{3}\right)\right]^2$$

$$= \left[\frac{7}{12}\right] - \left[\frac{7}{12}\right]^2$$

$$= \frac{84}{144} - \frac{49}{144} = \boxed{\frac{35}{144}}$$

(d)

$$Var[X \mid Y=1] = E[X^2 \mid Y=1] - \left[E[X \mid Y=1]\right]^2$$

$$= \sum_x x^2 \cdot f(x,1) - \left[\sum_x x \cdot f(x,1)\right]^2$$

$$= \left[(1)^2\left(\frac{1}{3}\right)\right] - \left[(1)\cdot\left(\frac{1}{3}\right)\right]^2$$

$$= \frac{1}{3} - \frac{1}{9} \boxed{= \frac{2}{9}}$$

(e)
$$E[X^3 + X^2 + 3Y^7 \mid Y=1]$$

$$= E[X^3 \mid Y=1] + E[X^2 \mid Y=1] + 3 \cdot E[Y^7 \mid Y=1]$$

$$= \sum_x x^3 \cdot f(x,1) + \sum_x x^2 \cdot f(x,1) + 3 \cdot (1)^7$$

$$= (1)^3\left(\frac{1}{3}\right) + (1)^2\left(\frac{1}{3}\right) + 3 = \frac{2}{3} + 3 \boxed{= \frac{11}{3}}$$

2.
$$\vec{v}_1 = [1,1,1], \quad \vec{v}_2 = [1,0,0]$$

$$p_1 = [3,3,3], \quad p_2 = [1,2,3], \quad p_3 = [0,0,1]$$

$$p_1 \Rightarrow \frac{\langle p_1, v_1\rangle}{\langle v_1, v_1\rangle} \cdot v_1 + \frac{\langle p_1, v_2\rangle}{\langle v_2, v_2\rangle} \cdot v_2$$

$$\Rightarrow \frac{9}{3} \cdot [1,1,1] + \frac{3}{1} \cdot [1,0,0] \boxed{= [6,3,3]}$$

$$p_2 \Rightarrow \frac{\langle p_2, v_1\rangle}{\langle v_1, v_1\rangle} \cdot v_1 + \frac{\langle p_2, v_2\rangle}{\langle v_2, v_2\rangle} \cdot v_2$$

$$\Rightarrow \frac{6}{3} \cdot [1,1,1] + \frac{1}{1} \cdot [1,0,0] \boxed{= [3,2,2]}$$

$$p_3 \Rightarrow \frac{\langle p_3, v_1\rangle}{\langle v_1, v_1\rangle} \cdot v_1 + \frac{\langle p_3, v_2\rangle}{\langle v_2, v_2\rangle} \cdot v_2$$

$$\Rightarrow \frac{1}{3} \cdot [1,1,1] + 0 \boxed{= \left[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right]}$$

3. $P(\text{Heads}) = \frac{2}{3}$

Since $n = 100 > 30$, we approximate using Central Limit Theorem.

$$\mu = (100)\left(\frac{2}{3}\right) = \frac{200}{3}$$

$$\sigma^2 = (100)\left(\frac{2}{3}\right)\left(\frac{1}{3}\right) = \frac{200}{9}$$

$$P(\text{Heads} \le 50) = P\left(\frac{\text{Heads} - \frac{200}{3}}{\sqrt{\frac{200}{9}}} \le \frac{50 - \frac{200}{3}}{\sqrt{\frac{200}{9}}}\right)$$

$$= P\left(\frac{\text{Heads} - \frac{200}{3}}{\sqrt{\frac{200}{9}}} \le \frac{-16.667}{4.71404}\right)$$

$$= P\left(\frac{\text{Heads} - \frac{200}{3}}{\sqrt{\frac{200}{9}}} \le -3.536\right)$$

$$= \Phi(-3.536) \boxed{\approx .00020}$$