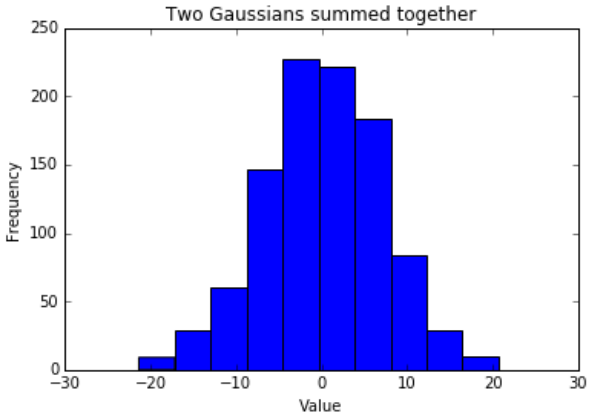
Daniel Diamont – dd28977

Lab 1

Q1)

a)



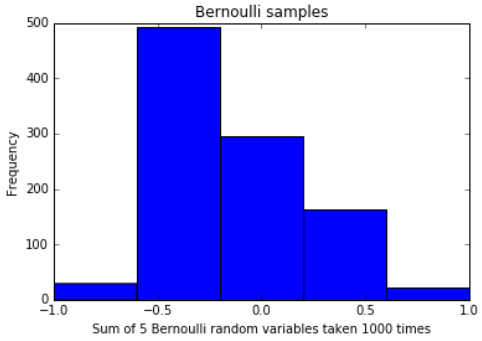

```
estimated mean of the new distribution is: 0.1448342891286605
estimated variance of the new distribution is: 48.29735278326488
```

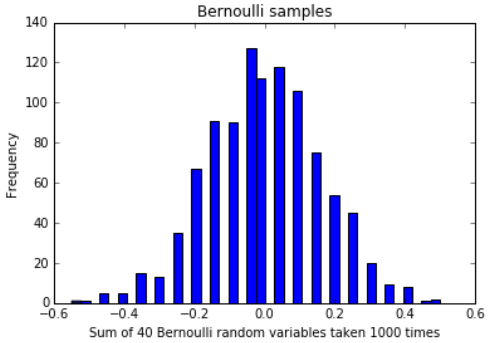
Note: Two Gaussians summed together makes a new Gaussian. The means add and the variances add.

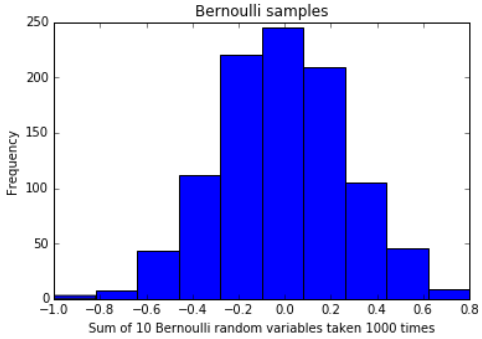
b)

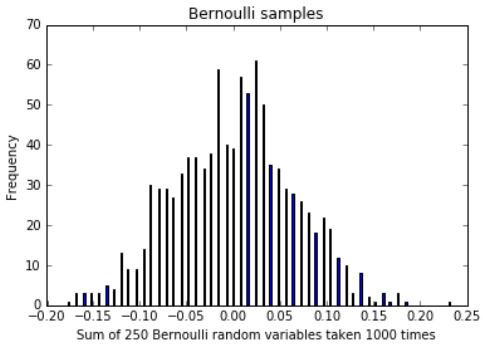
estimated mean: 0.1448342891286605
estimated variance: 48.29735278326488

Q2)









Q3)

```
# generate 25,000 samples from a gaussian distribution with mean 0 and std dev 5
norm_dist = np.random.normal(loc = 0, scale = 5, size = 25000)

# calculate estimated mean
mean_est = np.sum(norm_dist)/(norm_dist.size)
print("Estimated mean of distribution is: " + str(mean_est))

#calculate variance E[(X - E[X])^2]
x_minus_mean_sq = (norm_dist - mean_est)*(norm_dist - mean_est)
var_est = np.sum(x_minus_mean_sq)/(x_minus_mean_sq.size)

#calculate standard deviation as sqrt(variance)
std_dev = np.sqrt(var_est)

print("Estimated std dev of distribution is: " + str(std_dev))
```

```
Estimated mean of distribution is: -0.03111066484561935
Estimated std dev of distribution is: 4.9990729224585815
```

Q4)

```
#mean matrix
u = [-5,5]
#covariance matrix
covar = [[20, 0.8],[0.8,30]]

# generate 10,000 samples of 2D data from Gaussian distribution
d1,d2 = np.random.multivariate_normal(u, covar, 10000).T

#estimate the mean of the distribution
u1 = np.sum(d1)/d1.size
u2 = np.sum(d2)/d2.size
u_est = u1 + u2

#estimate the covariance matrix for the data
t1 = (d1-u1)*(d1-u1)
var1 = np.sum(t1)/t1.size #variance of distribution 1
t2 = (d2-u2)*(d2-u2)
var2 = np.sum(t2)/t2.size #variance of distribution 2

t_co = (d1-u1)*(d2-u2)
covar = np.sum(t_co)/t_co.size #covariance of distribution 1 and 2

cov = [] # covariance matrix
cov.append([var1,covar])
cov.append([covar,var2])

print("Estimated mean of the distribution: " + str(u_est))
print("Covariance matrix of the distribution: ")

df = pd.DataFrame(cov)
df.rename(index={0:'d1',1:'d2'}, inplace=True)
df.rename(columns={0:'d1',1:'d2'}, inplace=True)
display(df)
```

```
Estimated mean of the distribution: 0.04985467900170182
Covariance matrix of the distribution:
```

	d1	d2
d1	20.314510	0.808563
d2	0.808563	29.855643

Q5)

a)

```
data.shape
```

```
(452, 280)
```

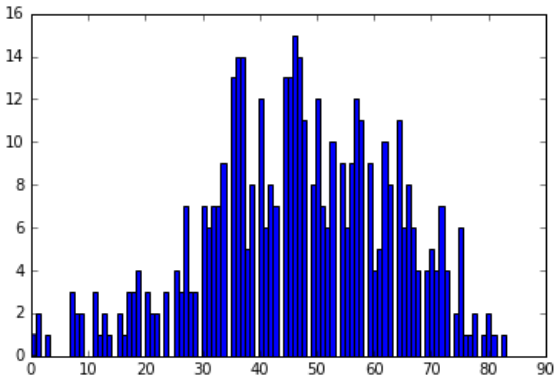
We can see that there are 451 rows and 280 columns referring to 451 patients with 280 features per patient

b)

Exploring column 0

```
data[0].describe()
```

```
count    452.000000
mean      46.471239
std       16.466631
min        0.000000
25%       36.000000
50%       47.000000
75%       58.000000
max       83.000000
Name: 0, dtype: float64
```

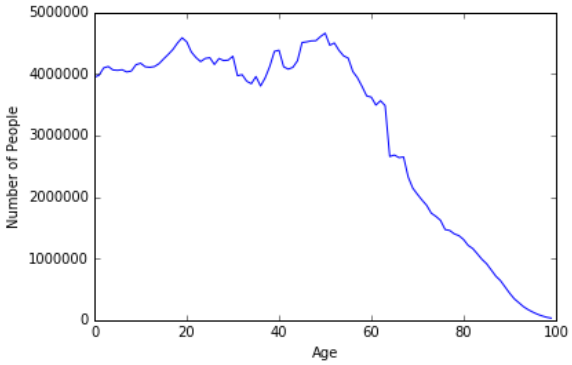


I suspect that this is patient age given that the mean is about 46.5 years of age. I suspect that the population of the U.S. has a mean within one standard deviation of this distribution. Additional notes supporting this assertion are that there are no outliers and no negative numbers.

Let's check US census data to see if this makes sense.

Link to dataset from US census: https://www.census.gov/data/tables/2017/demo/popest/nation-detail.html#par_textimage_98372960

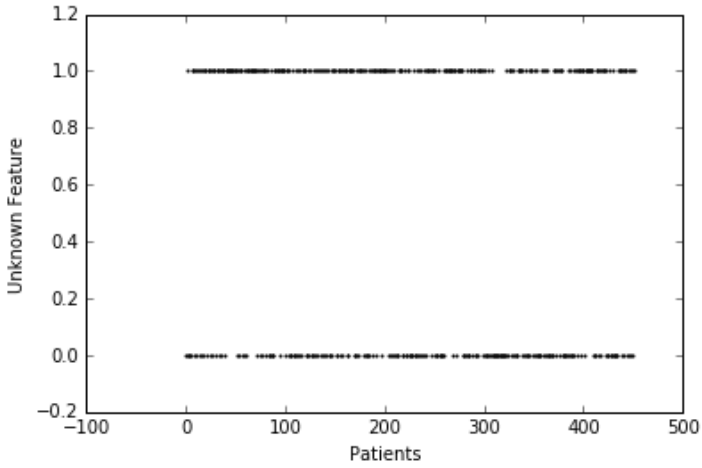
This dataset contains the age of people in the U.S. during the 2010 Census.




```
Mean age of population in US during 2010 census: 37.29410567546405
```

Because the mean of the US population is within the mean + standard deviation of feature 0, a good guess for feature 0 is patient age.

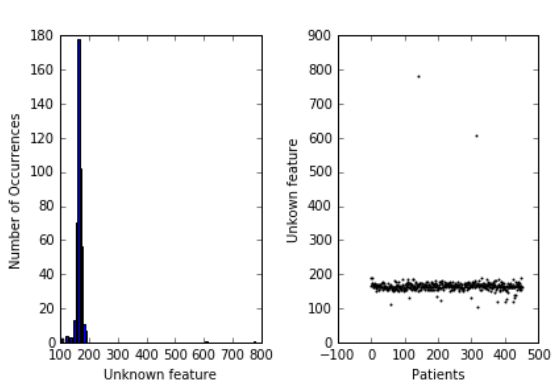
Now, to explore feature one

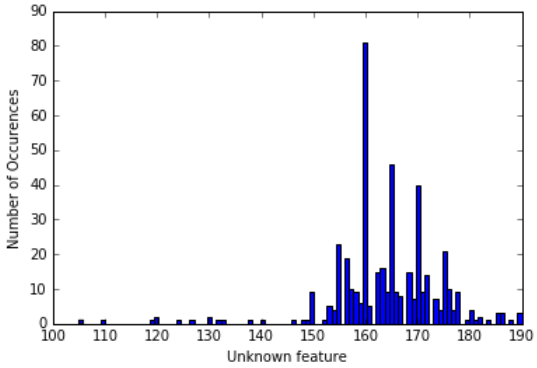


The data looks binary, and looks pretty equally distributed among 0's and 1's. Since these are patients, it is possible that feature 1 could be gender, as that one be a simple and relevant way to classify each patient for medical purposes.

Let's explore feature 2. Plotting a histogram and scatterplot for feature 2:

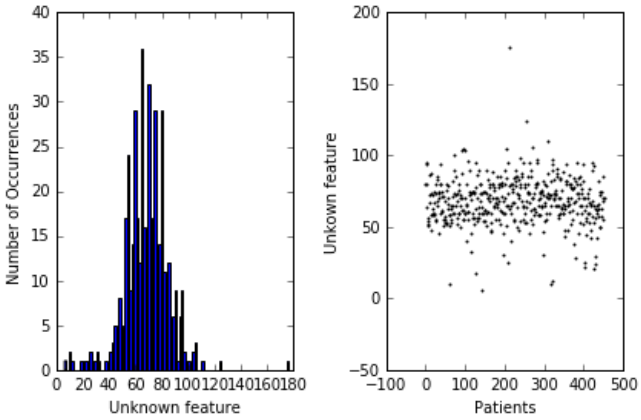
```
Mean: 166.18805309734512
```





The data looks Gaussian. If it is an attribute of a patient like height, weight, body mass index, etc, it would explain why the values are small for small ages, large for ages in the middle of the dataset, and smaller again for old ages.

Let's move on to feature 3 for now.



Data also looks Gaussian, but we are going to need some more help to decode what this data is. If I am right about feature 1 being gender, then perhaps we will see a difference in feature 2 and feature 3 with respect to gender.

```
          0          2          3
1
0  47.546798  171.315271  72.724138
1  45.594378  162.008032  64.457831
```

Looks like there is indeed a difference between the row 0 and row 1 with respect to columns 2 and 3. Considering body measurement metrics like height, weight, and BMI, it would make sense that the average values in these categories would be greater for males than the average values for women. Let's operate under the assumption that males are in row 0 and women are in row 1.

Now, let's try to identify the remaining two features. We can check for height and weight.

The National Center for Health Statistics has reported that the average height for an adult male (age 20 and over) is 69.2" (175.8 cm) and the average weight is 195.7 lbs (88.8 kg) For adult women, the average height is 63.7" (161.8 cm) and average weight is 168.5 lbs (76.43 kg).

Link: https://www.cdc.gov/nchs/fastats/body-measurements.htm

```
data filt adult = data filt[data filt[0] > 20] #only display ages 20+
```

	2								3							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
1																
0	185.0	170.875676	6.690414	152.0	166.0	170.0	175.0	190.0	185.0	75.740541	14.013492	18.0	68.0	75.0	82.0	176.0
1	233.0	160.115880	4.966210	146.0	156.0	160.0	163.0	174.0	233.0	66.145923	13.088399	42.0	56.0	64.0	72.0	124.0

We can see that for features 2 and 3, the average height and weight (from NCHS) for both men and women, lie within one standard deviation of the average height and weight for both men and women in our patient data. This suggests that indeed feature 2 is patient height in centimeters, and feature 3 is patient weight in kilograms.

c)

c) Replace missing values with average of corresponding feature column

```
data.fillna(data.mean()) # fill NaN values
```

	0	1	2	3	4	5	6	7	8	9	...	270	271	272	273	274	275	276	277	278	279
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	23.3	49.4	8
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	20.4	38.8	6
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	12.3	49.0	10
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	34.6	61.6	1

d)

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

One way to do this would be to calculate the correlation between each feature column with the label column. The columns that have a higher absolute value for the correlation are the columns which influence the patient condition the most.