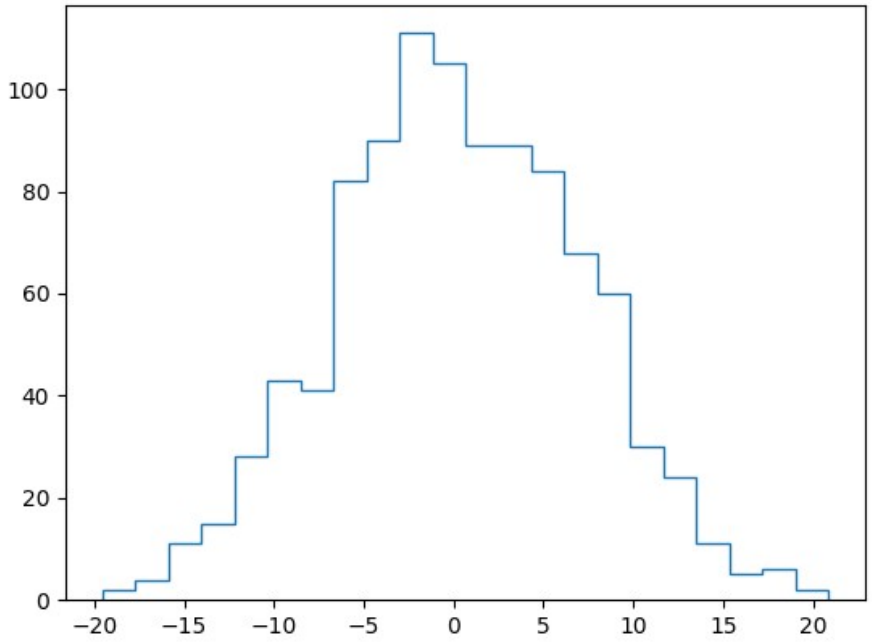
Hayden Prairie
Connie Wang
Drew Latz

Lab 1

Programming Problems

Problem 1



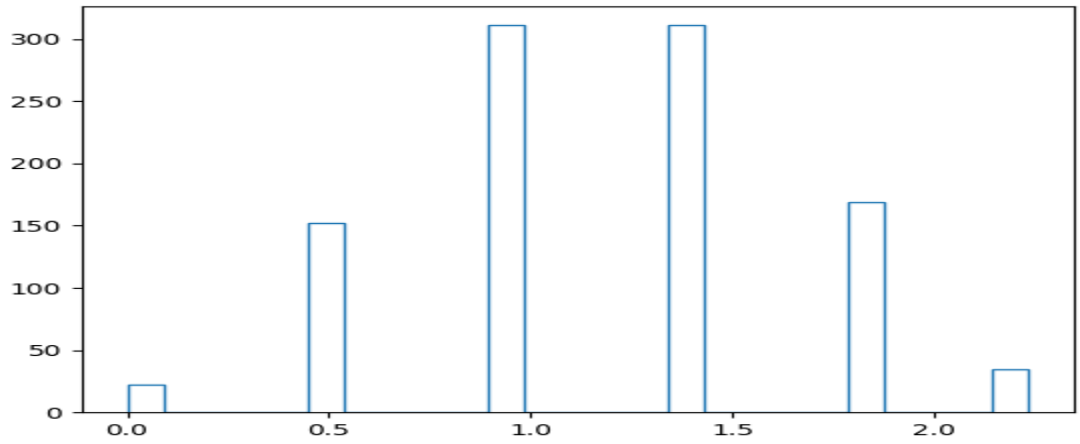

```
# Create Distributions
distribution1 = np.random.normal(-10,5,1000)
distribution2 = np.random.normal(10,5,1000)
finalDistribution = np.add(distribution1,distribution2)

# Plot Distributions
counts, bins = np.histogram(finalDistribution)
plt.stairs(counts,bins)
plt.show();

# Estimate variance and expectation
var = np.std(finalDistribution)
exp = np.average(finalDistribution)
print(var,exp)
```

The estimate expected value is 0.0028278275992278737 and the estimated standard deviations is 7.117456936519824.

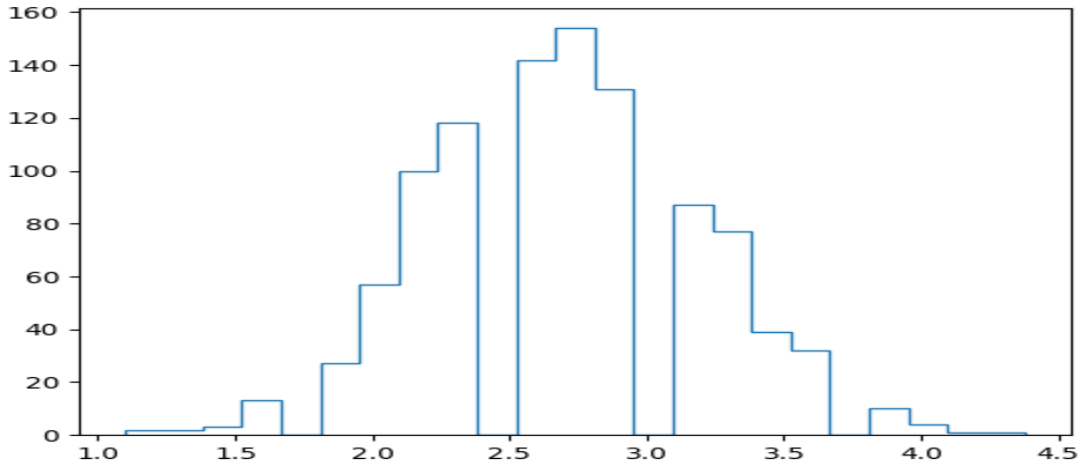
Hayden Prairie
Connie Wang
Drew Latz

Problem 2

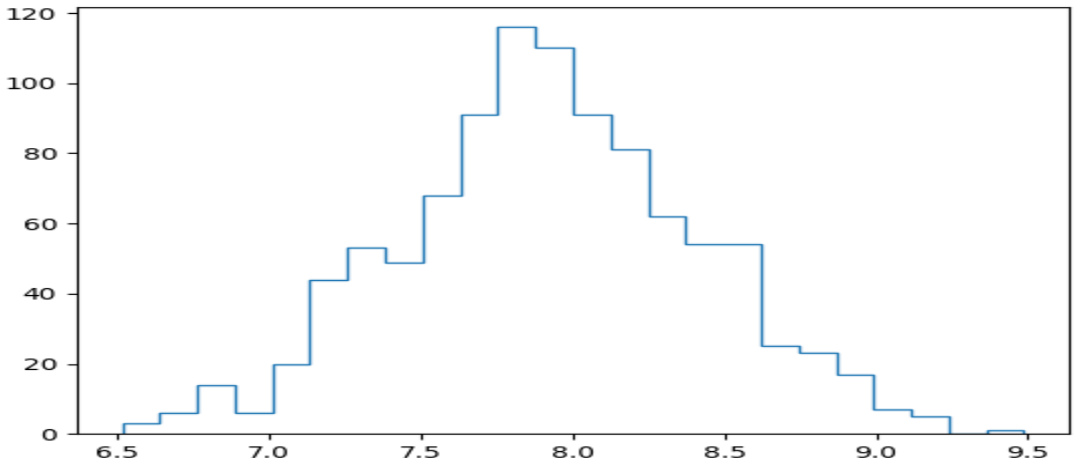
N = 5



N = 30



N = 250



Hayden Prairie
Connie Wang
Drew Latz

```
n= 250
final = []

for a in range(1000):
    temp = bernoulli.rvs(0.5, size=n)
    final.append((1/(math.sqrt(n)))*np.sum(temp))

counts, bins = np.histogram(final,bins='auto')
plt.stairs(counts,bins)
plt.show();
```

Hayden Prairie
Connie Wang
Drew Latz

Problem 3

Expectation = 0.024779456130474463

Variance = 5.040992624369163

```
dist = np.random.normal(0,5,25000)
E = np.sum(dist)/25000
Var = math.sqrt(np.sum(np.power(np.absolute(np.add(dist,-E)),2)) / 25000)

print(E,Var)
```

Hayden Prairie
Connie Wang
Drew Latz

Problem 4

From the estimated expectation matrix and covariance matrix we get the following:

```
(460J) hprairie@DESKTOP-CAT2SDB:~/460J/Labs/Lab1$ /home/hprairie/miniconda3/envs/460J/bin/python /home/hprairie/460J/Labs/Lab1/Problem4.py
[[-4.95618729]
 [ 4.92992777]]
[[20.22962247  0.8137144 ]
 [ 0.8137144  29.45294138]]
```

```python
dist = np.random.multivariate_normal([-5,5],[[20,.8],[.8,30]], 10000)
rows,col = dist.shape
finalExpectationMatrix = np.zeros(shape=(col,1))


for index in range(col):
    # Calculate the expectation and variance of speciifc axis
    expectation = np.sum(dist[:,index])/rows
    finalExpectationMatrix[index,0] = expectation

# Create the offset Matrix
difference = dist - np.transpose(finalExpectationMatrix)

# Calculate the Covariance Matrix
finalCovarianceMatrix = np.dot(np.transpose(difference), difference) / (rows-1)


print(finalExpectationMatrix)
print(finalCovarianceMatrix)
```

Hayden Prairie
Connie Wang
Drew Latz

Problem 5

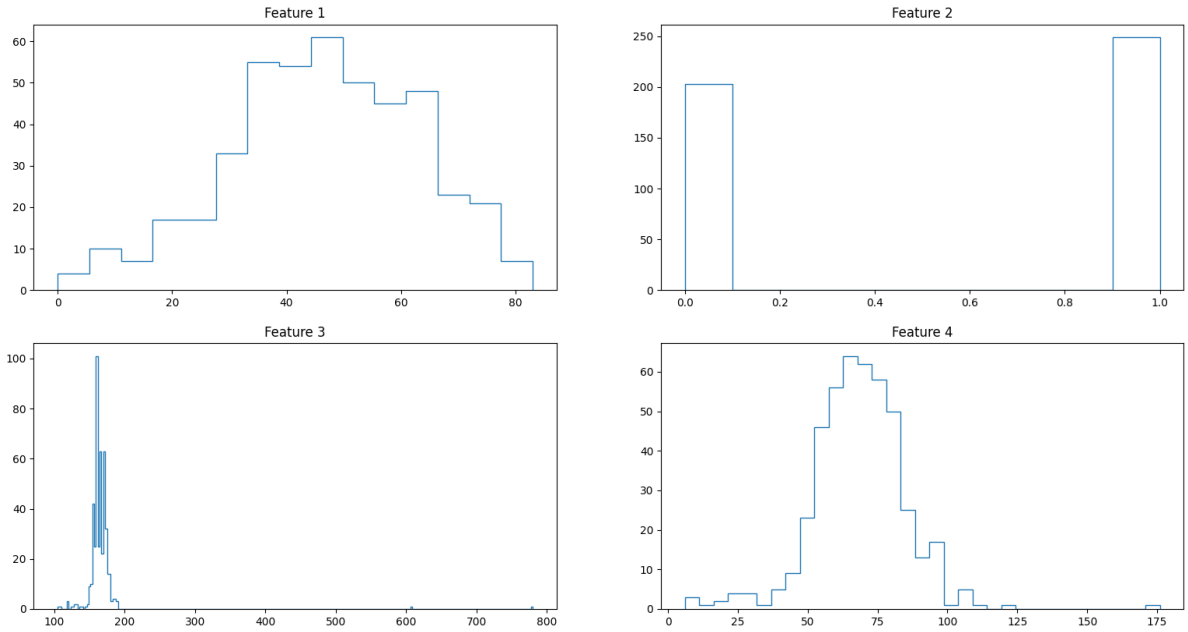
Part A

```
input = pd.read_csv("PatientData.csv", header = None)

# PART A determining patients and features
print(input.shape)
```

We get an output of (452,280) and thus have 482 patients and 279 features.

Part B

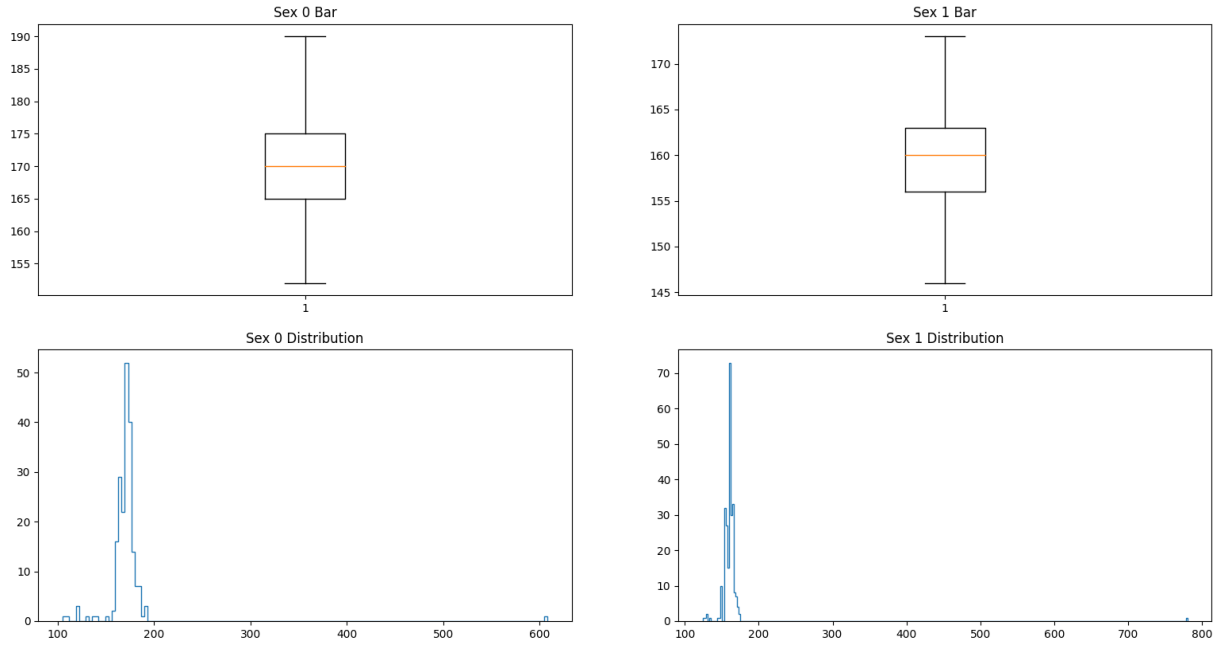
First we can start by plotting each of the different features in their own respective plots.



Instantly with the first feature being a normal distribution situation with nothing lower then 0 and nothing greater then 90, it appears to be the age of the patients. The second feature is either 0 or 1, which a good guess in terms of patient is sex.

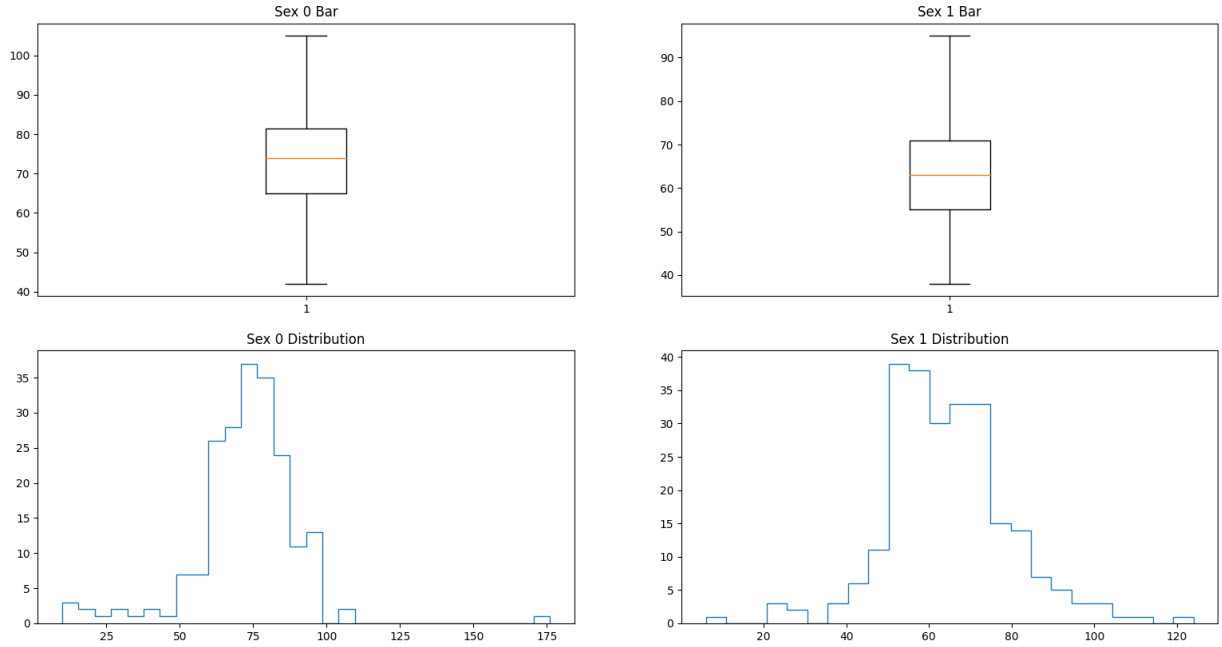
Given the assumtion that feature 2 indicates sex, then we can use this to help understand feature 3.

Hayden Prairie
Connie Wang
Drew Latz



If we ignore outliers to the data, we can see that the distribution are fairly normal and are offset based on sex. I good guess would be weight (ignoring outliers), which would align with the idea that one sex is slightly heavier then the other sex.

We can do the same thing for feature 4 and we get the following.



Hayden Prairie
Connie Wang
Drew Latz

Again the distribution for feature 4 based on assumed sex creates two normal distributions, that are slightly offset with estimate expected value. Based the numerical value and also the difference in expected values between the two sexs, a good guess would be height in inches.

Feature 1: Age

Feature 2: Sex

Feature 3: Weight (Pounds)

Feature 4: Height (Inches)

Part C

```
# Insert all missing data points with the average
input.replace("?",np.NaN,inplace=True)


# Cast all rows off object type to numeric values
input[input.columns[10]] = pd.to_numeric(input[input.columns[10]])
input[input.columns[11]] = pd.to_numeric(input[input.columns[11]])
input[input.columns[12]] = pd.to_numeric(input[input.columns[12]])
input[input.columns[13]] = pd.to_numeric(input[input.columns[13]])
input[input.columns[14]] = pd.to_numeric(input[input.columns[14]])

# Replace values of NaN with mean of the cols
input = input.to_numpy()
col_mean = np.nanmean(input, axis=0)
inds = np.where(np.isnan(input))
input[inds] = np.take(col_mean, inds[1])


# Export it back to a dataframe to export
input = pd.DataFrame(input)
input.to_csv("output.csv")
```

Part D

In order to pick out the most important features we can use logistic regression with a regularizer inorder to nullify non-important features. The lasso regularizer allows for good feature selection, and when fitting the data to a model with alpha=19.5, and then looking at the Beta values of the

Hayden Prairie
Connie Wang
Drew Latz

fitted model we can see that only features 5, 92, 247 have non-zero values and are likely to be the 3 most important features.

```
y_train = input.iloc[:,[-1]]
x_train = input.drop(y_train.columns,axis = 1)

lasso = Lasso(alpha=0)
lasso.fit(x_train,y_train)

print(lasso.sparse_coef_)
```

Hayden Prairie
Connie Wang
Drew Latz

Written Problems

Problem 1

a) ¼+⅛ = 5/12
b) ½
c) E[Y] = 0 (½) + 1 (½) = ½
   E[Y^2] = (0^2) (½) + (1^2) (½) = ½
   Var[Y] = E[Y^2] - E^2[Y] = ½ - ¼ = 1/4
d) E[X|Y=0] = 0 (½) + 1 (½) = ½
   E[X^2|Y=0] = (0^2) (½) + (1^2) (½) = ½
   Var[X|Y=0] = E[X^2|Y=0] - E^2[X|Y=0] = ½ - ¼ = 1/4
e) E[X^2-X^3+3Y^7|Y=1] = E[X^2|Y=1] - E[X^3|Y=1] + 3 E[Y^7|Y=1]
   E[X^2|Y=1] = (0^2) (1/3) + (1^2) (2/3) = ⅔
   E[X^3|Y=1] = (0^3) (1/3) + (1^3) (2/3) = ⅔
   E[X^2-X^3+3Y^7|Y=1] =⅔ -⅔ +3 (1) = 3
f) E[X^2+X|X] = { 0 when X = 0 and 2 when X = 1 }
g) E[Y|X] = { 0.4 when X = 0 and 4/7 when X = 1 }

Problem 2

Point 1: <p1,v1>/<v1,v1> v1 + <p1,v2>/<v2,v2> v2 =
$\frac{3+3+3}{1+1+1}$[1,1,1]+$\frac{3+0+0}{1+0+0}$[1,0,0] = [3,3,3]+[3,0,0] = [6,0,0]

Point 2: $\frac{1+2+3}{1+1+1}$[1,1,1]+$\frac{1+0+0}{1+0+0}$[1,0,0] = [2,2,2]+[1,0,0] = [3,2,2]

Point 3: $\frac{1}{1+1+1}$[1,1,1]+$\frac{0+0+0}{1+0+0}$[1,0,0] = [⅓,⅓,⅓]

Hayden Prairie
Connie Wang
Drew Latz

3.