ECE 460J Data Science Lab 1 Report:

Authors: William Wooten, Aksheeth Ilamparithi

Department of Electrical and Computer Engineering, The University of Texas at Austin

Programming Questions

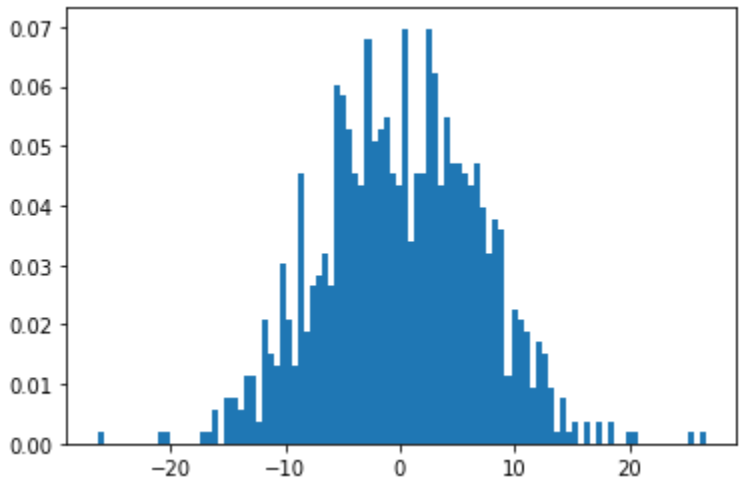
1. The following is the code used to derive the solutions:

```
# Authors: William, Aksheeth
# Assignment: Data Science Lab, Lab 1, Problem 1
import numpy as np
import matplotlib.pyplot as plt

def main():
   print("Problem 1 for 460J")
   mean_1 = - 10
   std_1 = 5
   samples_1 = np.random.normal(mean_1, std_1, 1000)
   mean_2 = 10
   std_2 = 5
   samples_2 = np.random.normal(mean_2, std_2, 1000)

   sums = [samples_1[i] + samples_2[i] for i in range(1000)]
   # plt.plot([i for i in range(-500, 500)], )
   plt.hist(sums, density=True, bins=100)

   print("For part A: We observe that the histogram is about symmetrical around
x = 0, and appears to also be a normal distribution\n we can see that the mean
is approximately zero")

   print("\nFor part B: Estimating Variance")
   mean = np.mean(sums)
   var = np.var(sums)
   print("The estimation of the mean is: " + str(mean))
   print("The estimation of the variance is: " + str(var))
   plt.show()


if __name__ == "__main__":
   main()
```

Which produced the following answers and output:

```
Problem 1 for 460J
For part A: We observe that the histogram is about symmetrical around x = 0, and
appears to also be a normal distribution
 we can see that the mean is approximately zero
```

```
For part B: Estimating Variance
The estimation of the mean is: 0.18017588300389906
The estimation of the variance is: 50.86183325327609
```

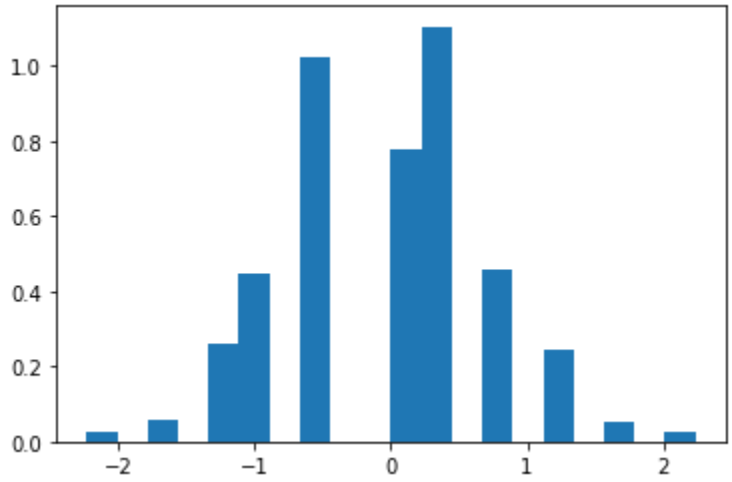


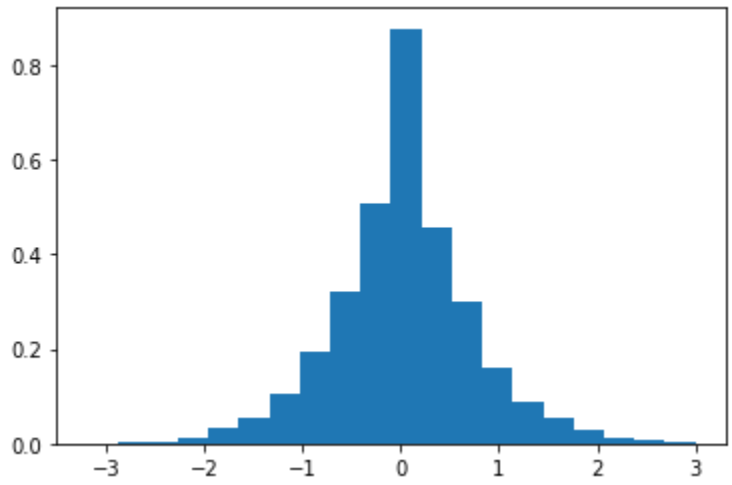
2. The following code was used to derive our solutions

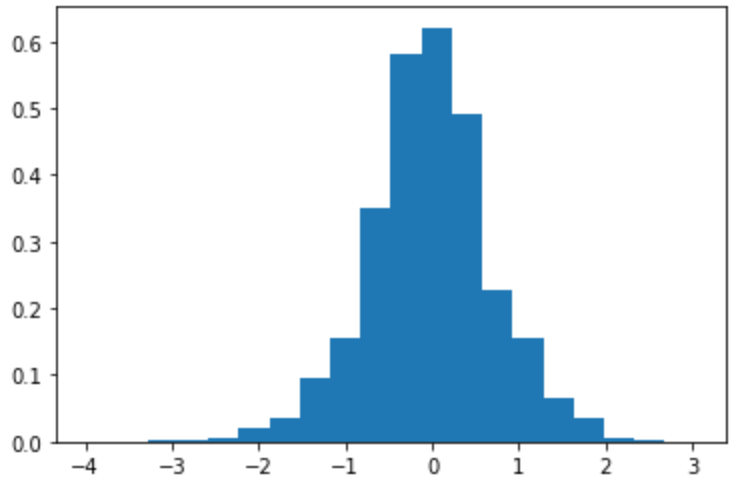
```python
# Authors: William, Aksheeth
# Assignment: Data Science Lab, Lab 1, Problem 2
import numpy as np
import math
import matplotlib.pyplot as plt
def z_distribution(n):
 z = []
 for i in range(0, 1000):
   # Compute a z_n
   z_n = 0
   for i in range(0, n):
     x = np.random.binomial(n = 1, p = .5, size = 1)
     if x == 0:
       z_n += -1
     else:
       z_n += 1
     z.append((1 / math.sqrt(n)) * z_n)
 plt.hist(z, density=True, bins=20)  # density=False would make counts
 plt.show()

z_distribution(n = 5)
z_distribution(n = 100)
```

```
z_distribution(n = 250)
```

Which produced the following graphs:







3. The following code was used to derive our solutions

```
# Authors: William, Aksheeth
# Assignment: Data Science Lab, Lab 1, Problem 3
import numpy as np
import math

samples = np.random.normal(0, 5, 25000)
mean  = sum(samples) / len(samples)
var = (1/(len(samples) - 1)) * sum((samples - mean) ** 2)
std = math.sqrt(var)

print("The approx. mean of the distribution is: ", mean)
print("The approximate standard deviation is ", std)
```

Which ran and produced the following output:

```
The approx. mean of the distribution is:  0.048462120727202684
The approximate standard deviation is  4.993659415168258
```

4. The following code was used to derive our solutions:

```
# Authors: William, Aksheeth
# Assignment: Data Science Lab, Lab 1, Problem 4
import numpy as np

means = [-5, 5]
cov = [[20, .8], [.8, 30]]
```

```
samples = np.random.multivariate_normal(means, cov, 10000)

est_mean = []
est_mean_x = 0
est_mean_y = 0
for sample in samples:
 est_mean_x += sample[0]
 est_mean_y += sample[1]
est_mean_x /= len(samples)
est_mean_y /= len(samples)
est_mean = [est_mean_x, est_mean_y]
print("Here is the vector estimating the means: \n" + str(est_mean))

cov = []
x = [sample[0] for sample in samples]
y = [sample[1] for sample in samples]
x_centered = [i - est_mean[0] for i in x]
y_centered = [i - est_mean[1] for i in y]
D_cent = []
for i in range(len(samples)):
 D_cent.append([x_centered[i], y_centered[i]])
D_cent = np.matrix(D_cent)
cov = np.multiply((1/(len(samples) - 1)), np.matmul(D_cent.transpose(),
D_cent))
print("\nThe Covariance Matrix: \n" + str(cov))
```

Which ran and produced the following answers:

```
Here is the vector estimating the means:
[-4.947086101688793, 5.008693824244377]

The Covariance Matrix:
[[20.22676744  0.86058423]
 [ 0.86058423 30.26848631]]
```

5. The following code was used to manipulate and explain PateintData.csv:

```
import pandas as pd
import math
def main():
   print("Solution For Problem 5, Homework 1, Data Science Lab")

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

   print("(a)\nThere are " + str(len(patient_data)) + " Patients in the
DataFrame")
   print("\nThere are " + str(len(patient_data.columns) - 1) + " Features in
the DataFrame")

   print("\n(b)\nThe first feature represents age in years")
   print("The second feature represents sex (0 for men, 1 for women)")
   print("The third feature represents height in centimeters")
   print("The fourth feature represents weight in kg")

   print("\n(c)\nYes there are missing values")

   # Replace each question mark with the average for each feature
   col = 1
   for label, feature in patient_data.iteritems():
       # Compute the avg
       avg = 0
       for value in feature.iteritems():
           if value[1] != "?":
               avg += float(value[1])
       avg /= len(feature)
       patient_data[label] = feature.replace(to_replace = "?", value = avg)

   # Compute the correlation coefficient
   print("\n(d)\nWe can determine which features influence the patient
condition by computing \nthe correlation Coeff between each individual feature
and the result and finding the ones closest to -1 and 1")

   corr = patient_data.corr()
   coeff = []
   for pair in corr[len(patient_data.columns) - 1].iteritems():
       if not math.isnan(pair[1]):
           coeff.append((pair[0], abs(pair[1])))

   coeff.sort(key = lambda x: x[1])
   coeff.reverse()
   # Record the most useful
   print("\nThe following indices represent the 3 features that are most
```

```
correlated with patient results:")
   for i in range(1, 4):
       print(coeff[i][0], end = " ")
   print("\nNote: we ignore the first one, because the most correlated value
computed with the result was itself")
if __name__ == "__main__":
   main()
```

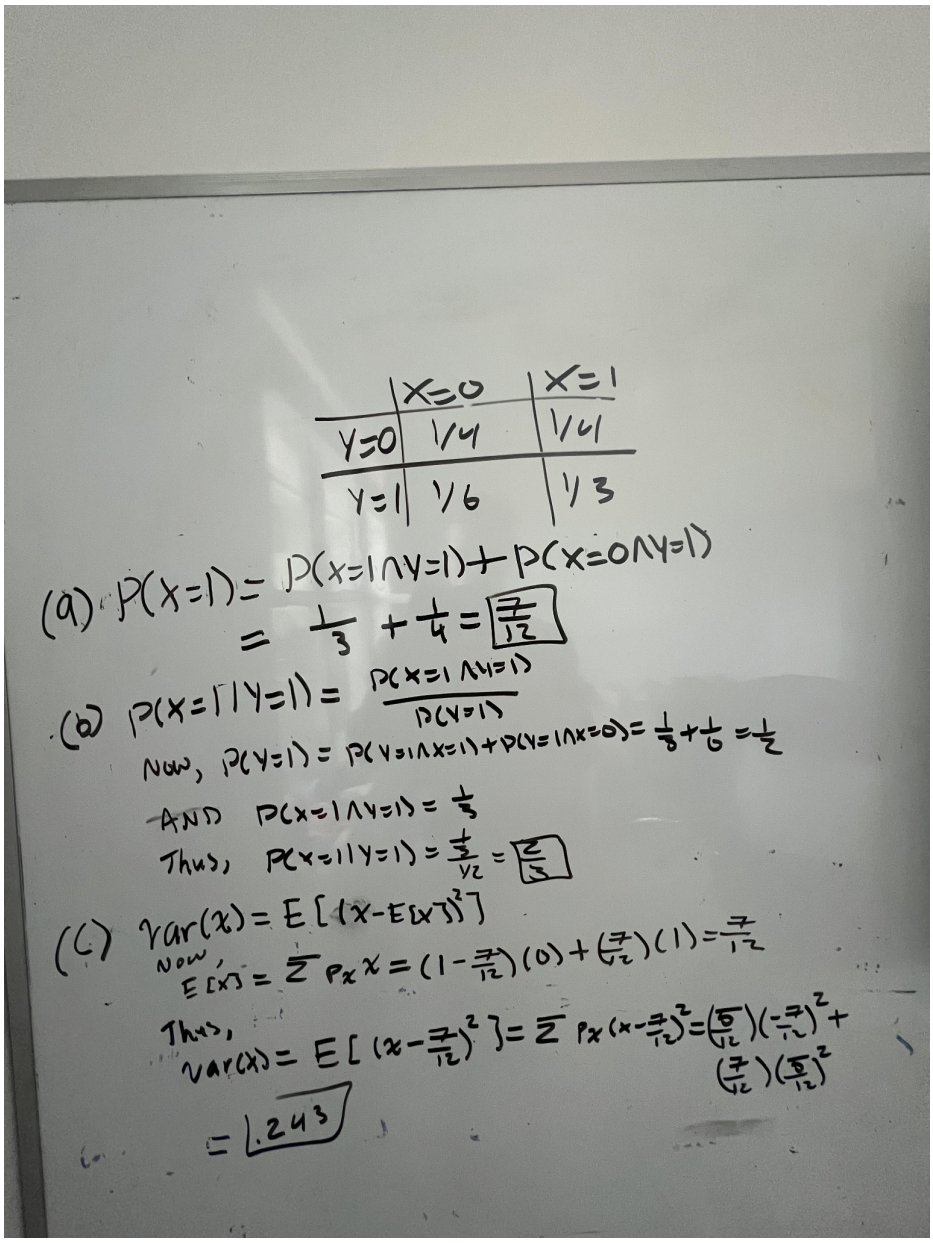
Which ran and produced the following outputs and answers:

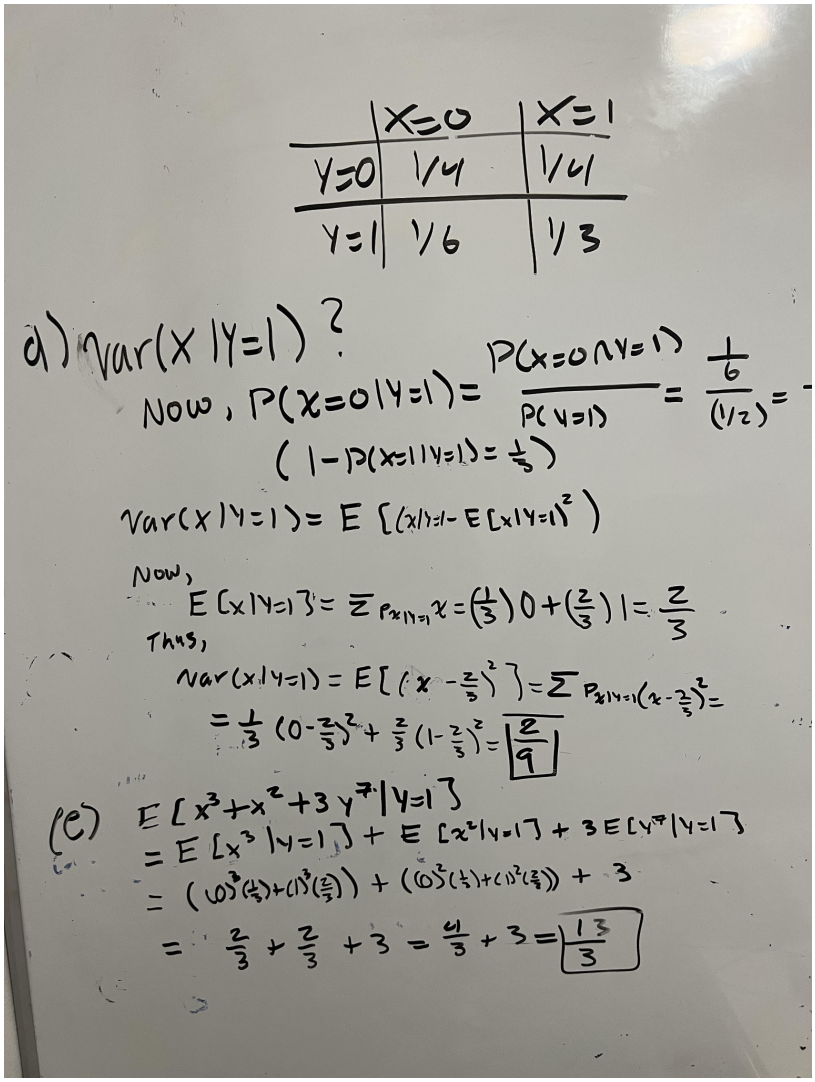
Solution For Problem 5, Homework 1, Data Science Lab

```
(a)
There are 452 Patients in the DataFrame

There are 279 Features in the DataFrame

(b)
The first feature represents age in years
The second feature represents sex (0 for men, 1 for women)
The third feature represents height in centimeters
The fourth feature represents weight in kg

(c)
Yes there are missing values

(d)
We can determine which features influence the patient condition by computing
the correlation Coeff between each individual feature and the result and finding
the ones closest to -1 and 1

The following indices represent the 3 features that are most correlated with
patient results:
90 4 92
Note: we ignore the first one, because the most correlated value computed with the
result was itself
```

Written Questions

1. The following work represents the evaluation and answers for the first written question





2. The following code represents the evaluation and answers for the second written question

```
# Calculate projections
import numpy as np

v1 = [1,1,1]
v2 = [1,0,0]

p1 = [3,3,3]
p2 = [1,2,3]
p3 = [0,0,1]

X = [v1, v2]

def calc_beta(y, X):
 X = np.matrix(X)
 return np.matmul(np.matmul(np.linalg.inv(np.matmul(X, X.transpose())), X),
np.matrix(y).transpose() )
```

```
def get_coords(y,X):
 return np.matmul(X.transpose(), y)

print(get_coords(calc_beta(p1, X),np.matrix(X)).transpose())
print(get_coords(calc_beta(p2, X),np.matrix(X)).transpose())
print(get_coords(calc_beta(p3, X),np.matrix(X)).transpose())
```

Which produced the following results:

```
[[3. 3. 3.]]
[[1.  2.5 2.5]]
[[5.55111512e-17 5.00000000e-01 5.00000000e-01]]
```

```
Note: Each vector represents the projection of p1, p2, and p3 (in that order ) onto the subspace [v1, v2]
```

3. The following work represents the evaluation and answers for the third written question