Keshav Shankar
kes298
ECE 1395
Homework 1

1) I could use regression to predict tumor size in radiology from medical images.

a) Some features I would use are pixel intensity values, shape features, and texture features.

b) For labels, I would use tumor size and tumor volume

c) For data collection, I would find hospitals' datasets of radiology medical images and corresponding measurements in them.

d) Some challenges are medical images are very noisy, the labeled data may have alot of variance, and biological systems are complex to predict.

2) I could use classification to classify if a patient has a specific disease.

a) Some features I would use are results from medical diagnostic tests, lab tests, and imaging studies, specifically blood pressure, heart rate, body temp, and respitory rate.

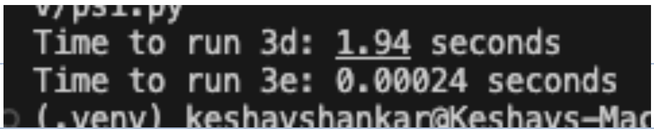
b) For labels, I would have either "has disease" or "no disease".

c) For data, I would get data sets from hospitals and treatment centers.

d) The data collected could all be from one place, which causes an imbalance and may not be diverse enough.

3c) Yes, the histogram for x does look like a gaussian distribution, and the histogram for z looks like a uniform distribution.

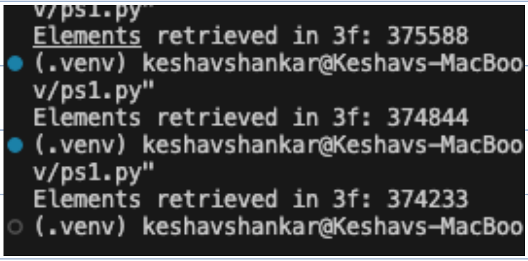
3e)



```
v/ps1.py
Time to run 3d: 1.94 seconds
Time to run 3e: 0.00024 seconds
(.venv) keshavshankar@Keshavs-Mac
```

The Second way in 3e is more efficient, because numpy has a built in override operator to perform such a calculation.

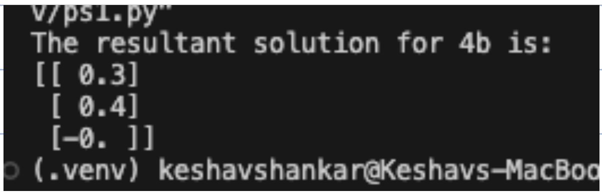
3f)



```
v/ps1.py"
Elements retrieved in 3f: 375588
(.venv) keshavshankar@Keshavs-MacBoo
v/ps1.py"
Elements retrieved in 3f: 374844
(.venv) keshavshankar@Keshavs-MacBoo
v/ps1.py"
Elements retrieved in 3f: 374233
(.venv) keshavshankar@Keshavs-MacBoo
```

Yes, there is a difference. This is because we are generating random numbers for the "z" vector, thus the # of values btwn. 0 & 1.5 will always change.

4b)

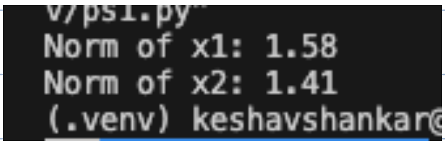


```
v/ps1.py"
The resultant solution for 4b is:
[[ 0.3]
 [ 0.4]
 [-0. ]]
(.venv) keshavshankar@Keshavs-MacBoo
```

4c)

$$x_1 = [-0.5 \quad 0 \quad 1.5] \Rightarrow (-0.5, 0, 1.5)^2 = (0.25, 0, 2.25)$$
$$= \sqrt{2.5} = \boxed{1.58} \quad \text{L2 norm}$$

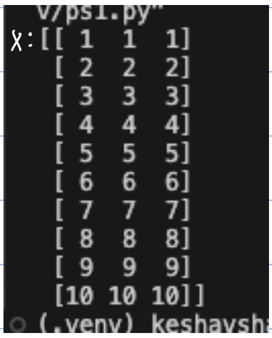
$$x_2 = [-1 \quad -1 \quad 0] \Rightarrow (-1, -1, 0)^2 = (1, 1, 0)$$
$$= \sqrt{2} = \boxed{1.41} \quad \text{L2 norm}$$



```
v/ps1.py"
Norm of x1: 1.58
Norm of x2: 1.41
(.venv) keshavshankar@
```

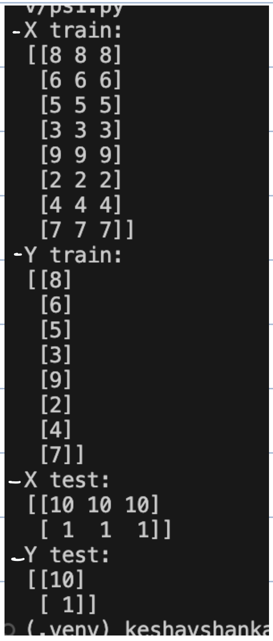
The python solution verifies that the calculations are correct.

5a)



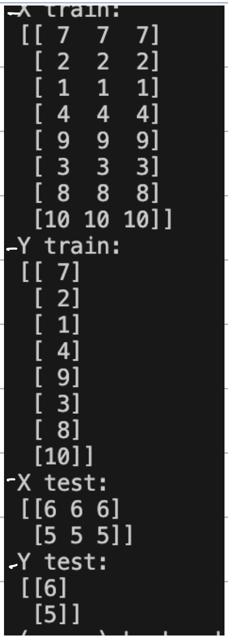
```
v/ps1.py"
X:[[ 1  1  1]
  [ 2  2  2]
  [ 3  3  3]
  [ 4  4  4]
  [ 5  5  5]
  [ 6  6  6]
  [ 7  7  7]
  [ 8  8  8]
  [ 9  9  9]
  [10 10 10]]
(.venv) keshavsh
```

5d)



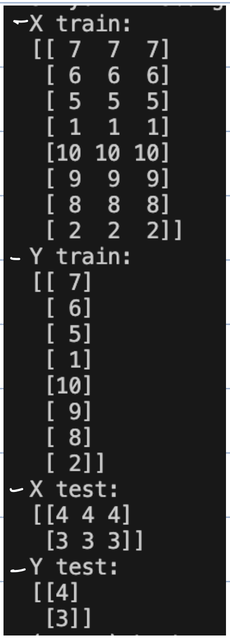
```
v/ps1.py
-X train:
 [[8 8 8]
  [6 6 6]
  [5 5 5]
  [3 3 3]
  [9 9 9]
  [2 2 2]
  [4 4 4]
  [7 7 7]]
-Y train:
 [[8]
  [6]
  [5]
  [3]
  [9]
  [2]
  [4]
  [7]]
-X test:
 [[10 10 10]
  [ 1  1  1]]
-Y test:
 [[10]
  [ 1]]
(.venv) keshavshanka
```

trial 1



```
-X train:
 [[ 7  7  7]
  [ 2  2  2]
  [ 1  1  1]
  [ 4  4  4]
  [ 9  9  9]
  [ 3  3  3]
  [ 8  8  8]
  [10 10 10]]
-Y train:
 [[ 7]
  [ 2]
  [ 1]
  [ 4]
  [ 9]
  [ 3]
  [ 8]
  [10]]
-X test:
 [[6 6 6]
  [5 5 5]]
-Y test:
 [[6]
  [5]]
```

trial 2



```
-X train:
 [[ 7  7  7]
  [ 6  6  6]
  [ 5  5  5]
  [ 1  1  1]
  [10 10 10]
  [ 9  9  9]
  [ 8  8  8]
  [ 2  2  2]]
- Y train:
 [[ 7]
  [ 6]
  [ 5]
  [ 1]
  [10]
  [ 9]
  [ 8]
  [ 2]]
- X test:
 [[4 4 4]
  [3 3 3]]
- Y test:
 [[4]
  [3]]
```

trial 3

No, you do not get the same submatricies every time because you are using a random seed to shuffle the arrays every time.