EE325 Assignment

Anoushka Dey 210010010
Savaliya Abhishek 21D070065
Ritesh Bahl 21D070057

August 10, 2022

1 Question 1

- How to select K students to collect data

  We can use ony one of the three given options to select k students.However we will write code for each option and will try to identify which option is the best.

- How the guess improve as function of k? And hence, what is a good k?

  as k increases, the average of data of k students will get more close to actual average value of all 10,000 students hence guess improves as k increases. So a good k is the max possible value of k which fits into compney's budget.

1.1 part a

Here are codes for options a,b and c of choosing k students. Each code takes k as input and gives scatter plot as output.

option (a)

```
import matplotlib.pyplot as plt
import numpy as np
import statistics
from statistics import mean

"""
In part (a) we ask k students as soon as we entered the campus
So, we will take first k entries of given data and find their average
"""
#first we will define an average function
def Average(l):
    avg = mean(l)
    return avg
#storing given data in an array called 'lines'
```

```
text_file = open("hw1a.txt", "r")
lines = text_file.readlines()
text_file.close()

"""
We are going to repeat experiment 50 times so we will get 50 average values
so for x axis we define an array of 50 elements 1 to 50
"""
a=list(range(50))
#now we will take k as input
k=int(input("Enter no of samples:"))
"""
Note that lines array will have elements of data type string
We must covert it to float data type in order to plot them on graph
So we will define a new array called 'num' which will store all elements of
'lines' in float data type.
"""
num = []
for i in range(len(lines)):
    t = float(lines[i])
    num.append(t)
"""
defining array 'average' which will store 50 average values
found in 50 repeatations of experiment
"""
average=list(range(50))

for i in range(50):
  x = num[:k]              #making an array of fist k elements
  average[i] = Average(x)  #taking their average
"""
before plotting these 2 lines will give actual values of average
and standerd deviation of entire data so that we can compare them later
"""
print(Average(num))
print((statistics.stdev(num)))
#now plot average values found in 50 experiments
plt.scatter(a, average)
plt.show()
"""
you will notice that since in all experiments we took same initial k values,
we got same average value in all 50 repeatations
but this value will be different for different values of k
"""
```

option (b)

```
import random
import matplotlib.pyplot as plt
import numpy as np
import statistics
from statistics import mean

"""
In part (b) we ask k students from an arbitary point
So, we will find an arbitary point from data and then take k samples
next to that point and find their average
"""
#first we will define an average function

def Average(l):
    avg = mean(l)
    return avg
#storing given data in an array called 'lines'
text_file = open("hw1a.txt", "r")
lines = text_file.readlines()
text_file.close()

"""
We are going to repeat experiment 50 times so we will get 50 average values
so for x axis we define an array of 50 elements 1 to 50
"""
y=list(range(50))
#now we will take k as input
k=int(input("Enter no of samples:"))
"""
Note that lines array will have elements of data type string
We must covert it to float data type in order to plot them on graph
So we will define a new array called 'num' which will store all elements of
'lines' in float data type.
"""
num = []
for i in range(len(lines)):
    t = float(lines[i])
    num.append(t)
#defining array 'average' which will store 50 average values found in
50 repeatations of experiment
average=list(range(50))

for i in range(50):
  z=random.choice(list(range(10000-k)))  #taking a random value
  x = num[z:z+k]                          #storing next k samples in an array
  average[i] = Average(x)                 #taking these k samples' average

print(Average(num))
print((statistics.stdev(num)))
plt.scatter(y, average)
plt.show()
```

option (c)

```
import matplotlib.pyplot as plt
import numpy as np
import statistics
from statistics import mean
"""
In part (c) we choose k students randomly
So, we will randomly select k samples from data
"""
#first we will define an average function
def Average(l):
    avg = mean(l)
    return avg
#storing given data in an array called 'lines'
text_file = open("hw1a.txt", "r")
lines = text_file.readlines()
text_file.close()
"""
We are going to repeat experiment 50 times so we will get 50 average values
so for x axis we define an array of 50 elements 1 to 50
"""
y=list(range(50))

k=int(input("Enter no of samples:"))
"""
Note that lines array will have elements of data type string
We must covert it to float data type in order to plot them on graph
So we will define a new array called 'num' which will store all elements of
'lines' in float data type.
"""
num = []
for i in range(len(lines)):
    t = float(lines[i])
    num.append(t)

average=list(range(50))
```

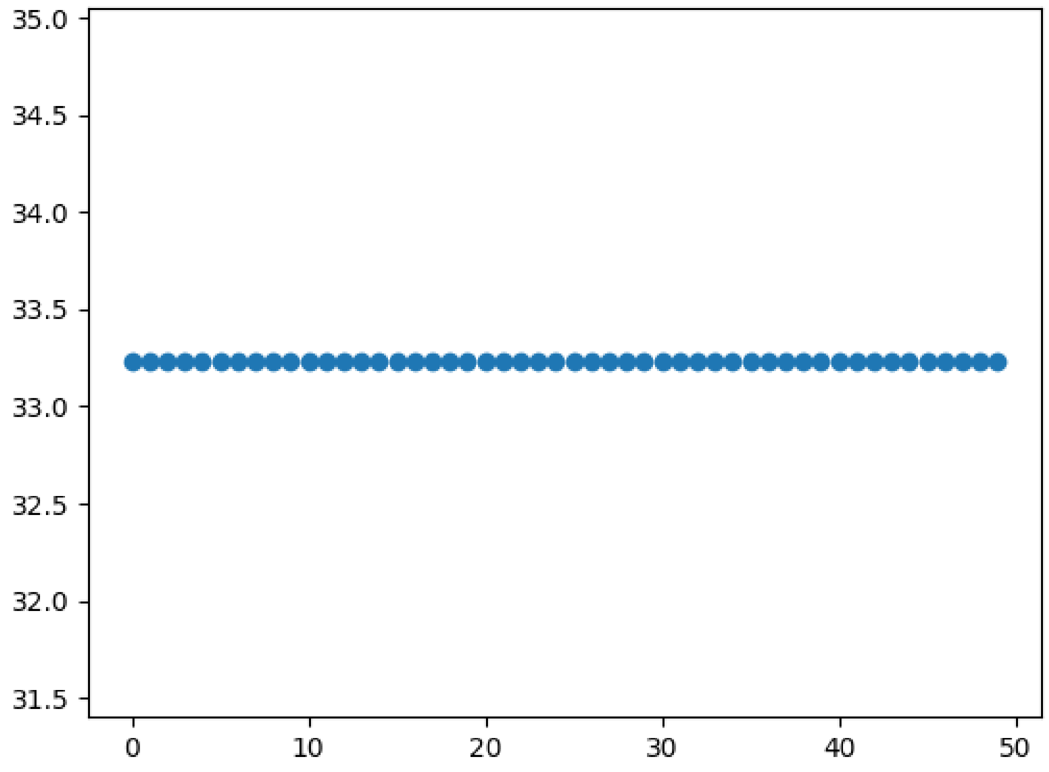

Figure 1: For k=10 in option a

```
for i in range(50):
  x = np.random.choice(num, size=k, replace=False) #selecting k random values
  average[i] = Average(x)                          #taking average

print(Average(num))
print((statistics.stdev(num)))
plt.scatter(y, average)
plt.show()
```

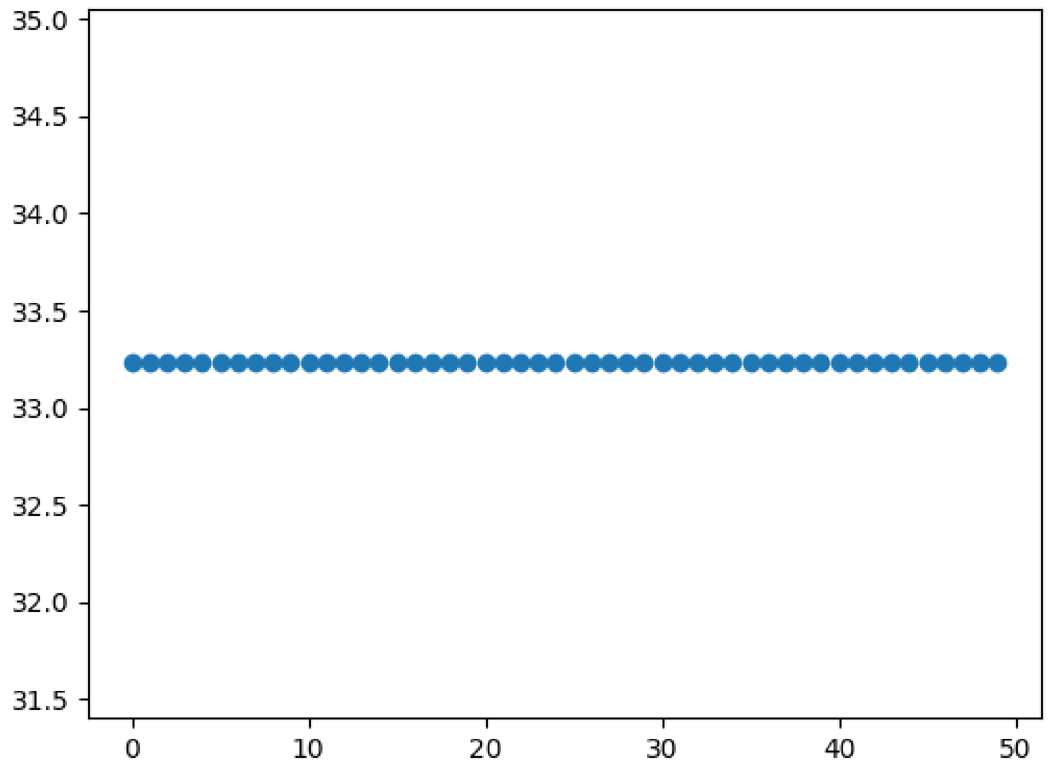

Figure 2: For k=20 in option a

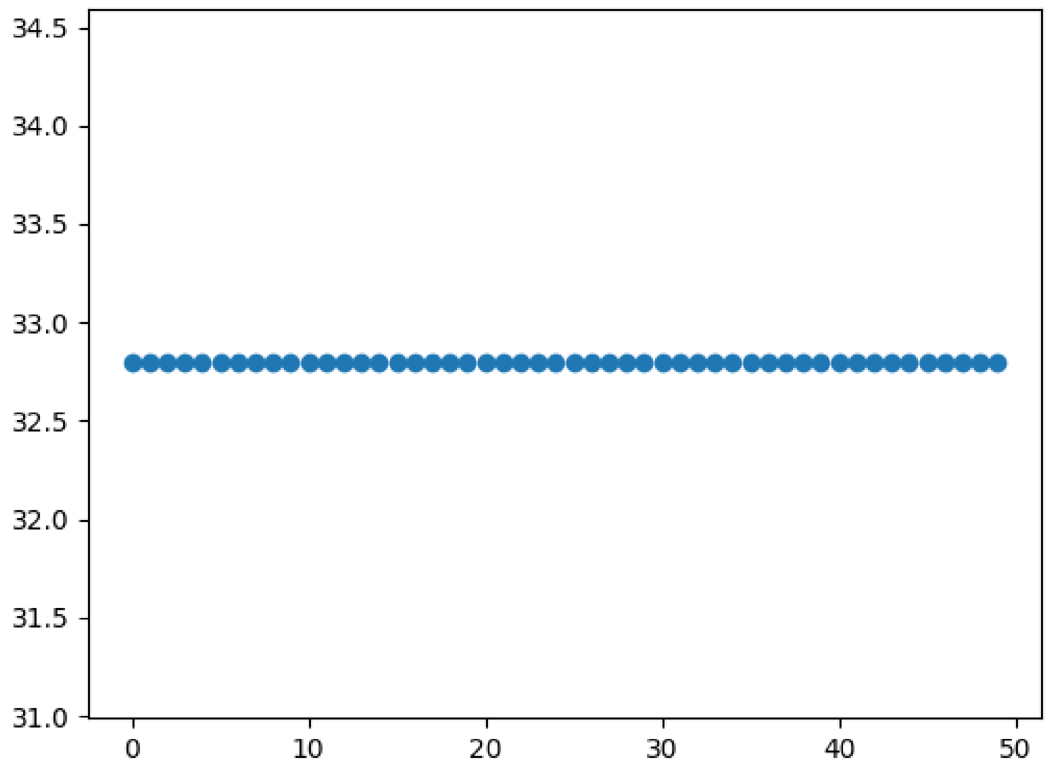

Figure 3: For k=50 in option a

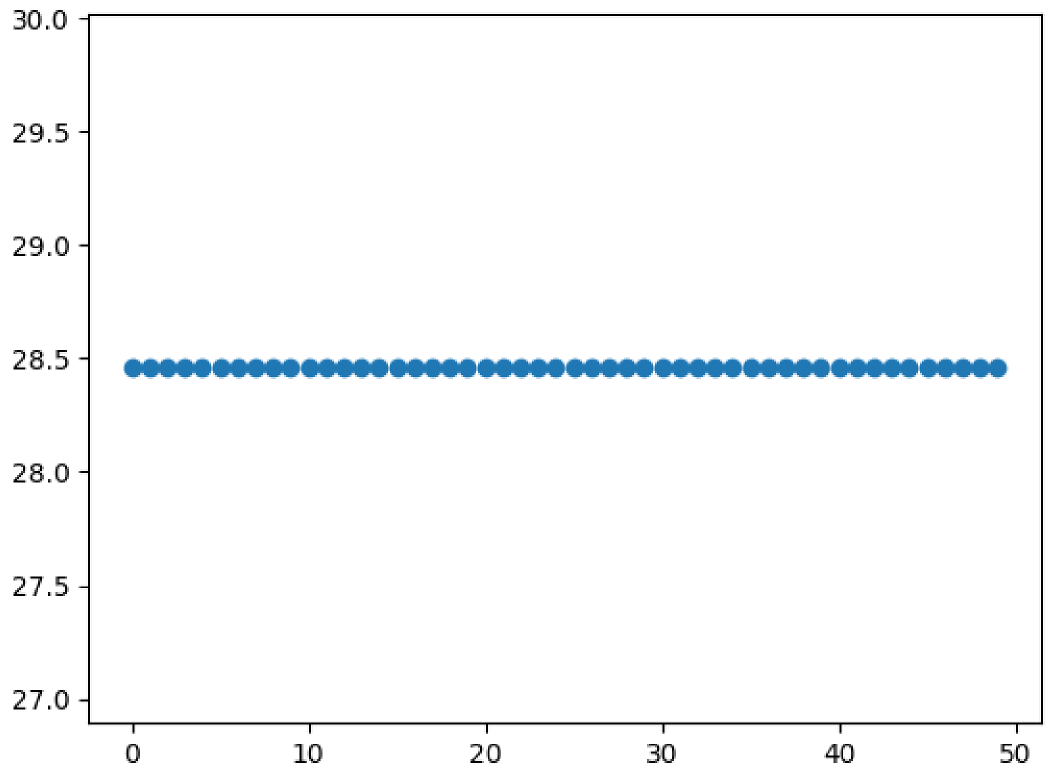

Figure 4: For k=100 in option a

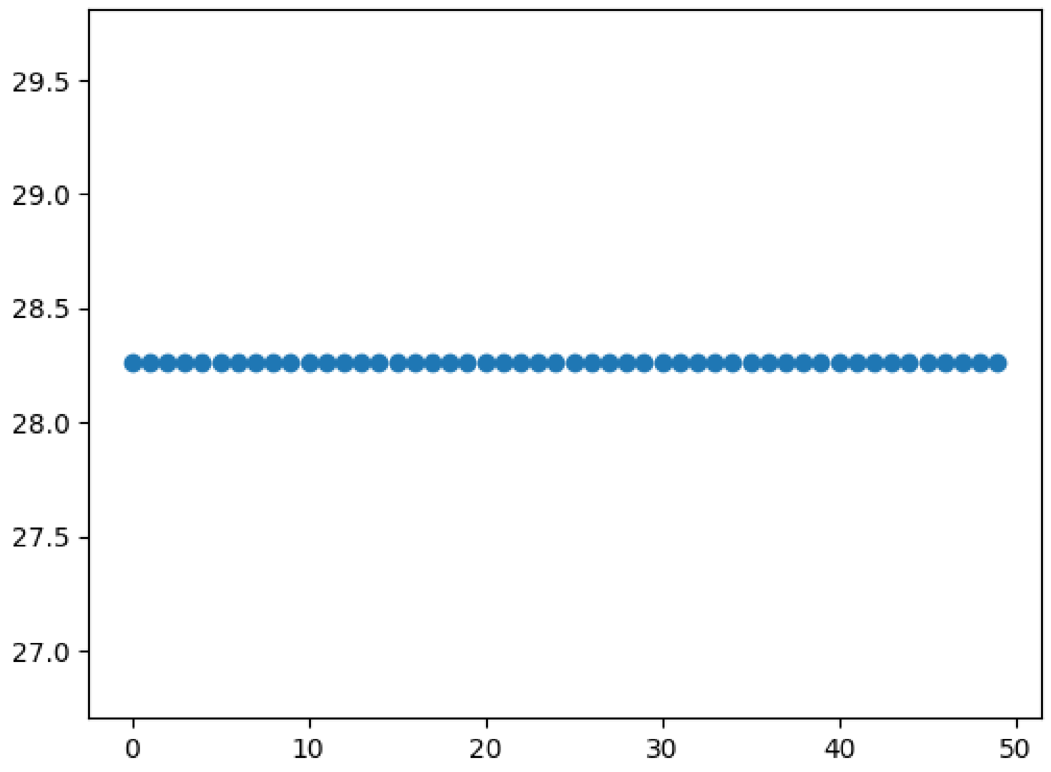

Figure 5: For k=200 in option a

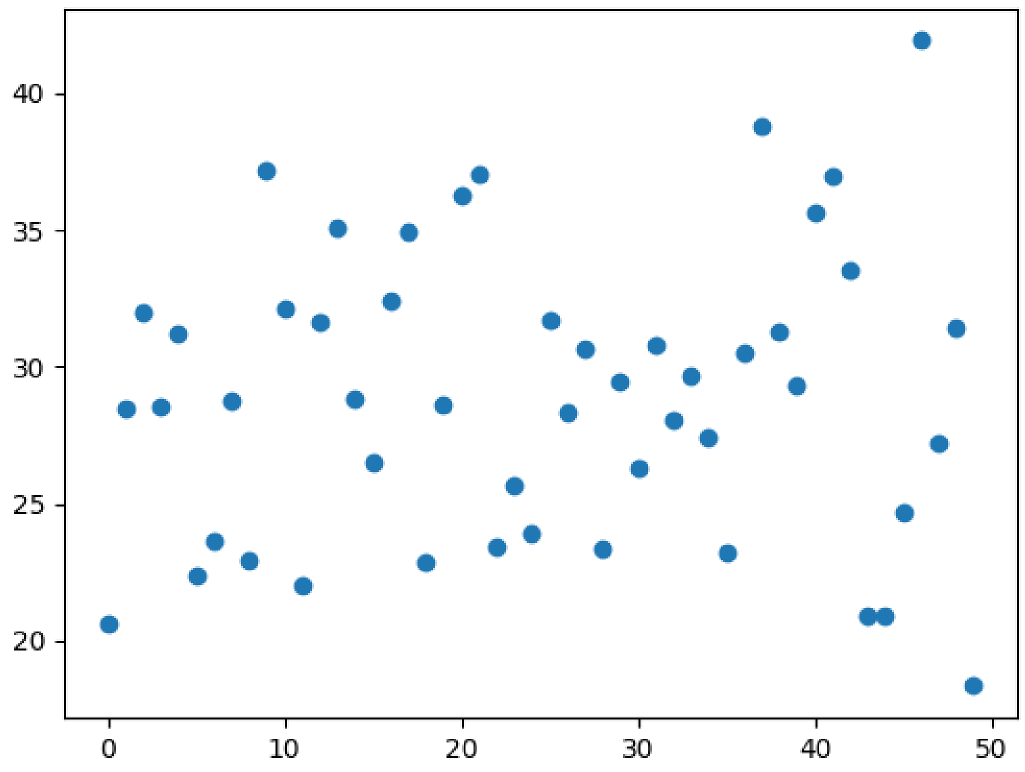

Figure 6: For k=10 in option b

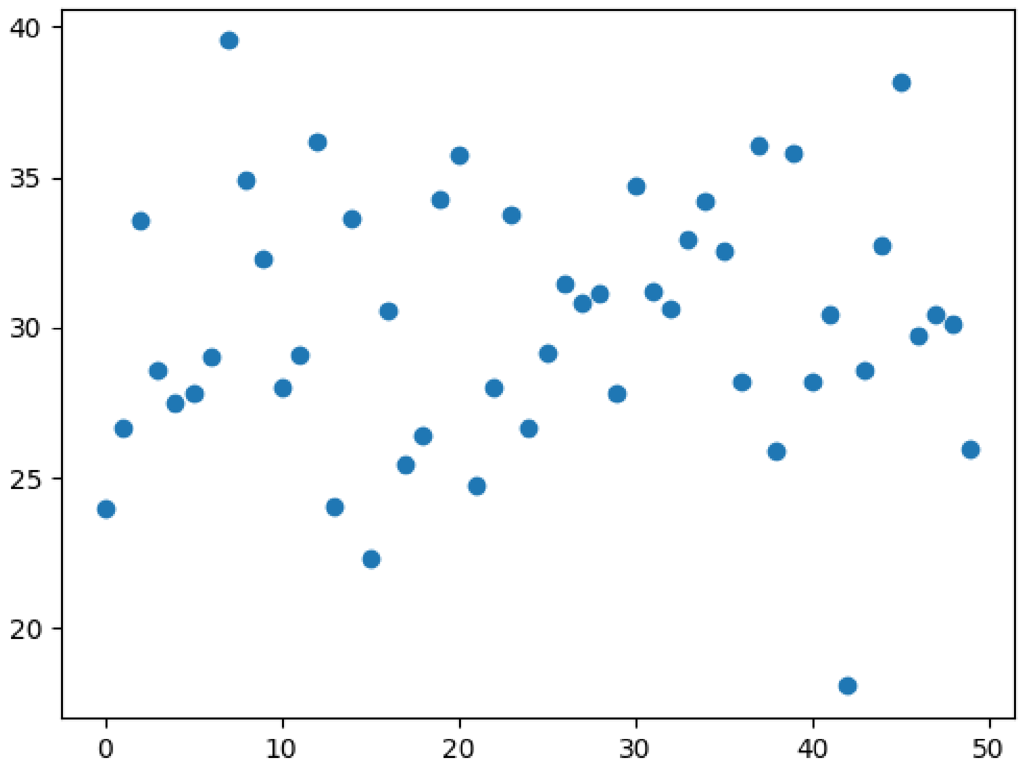

Figure 7: For k=20 in option b

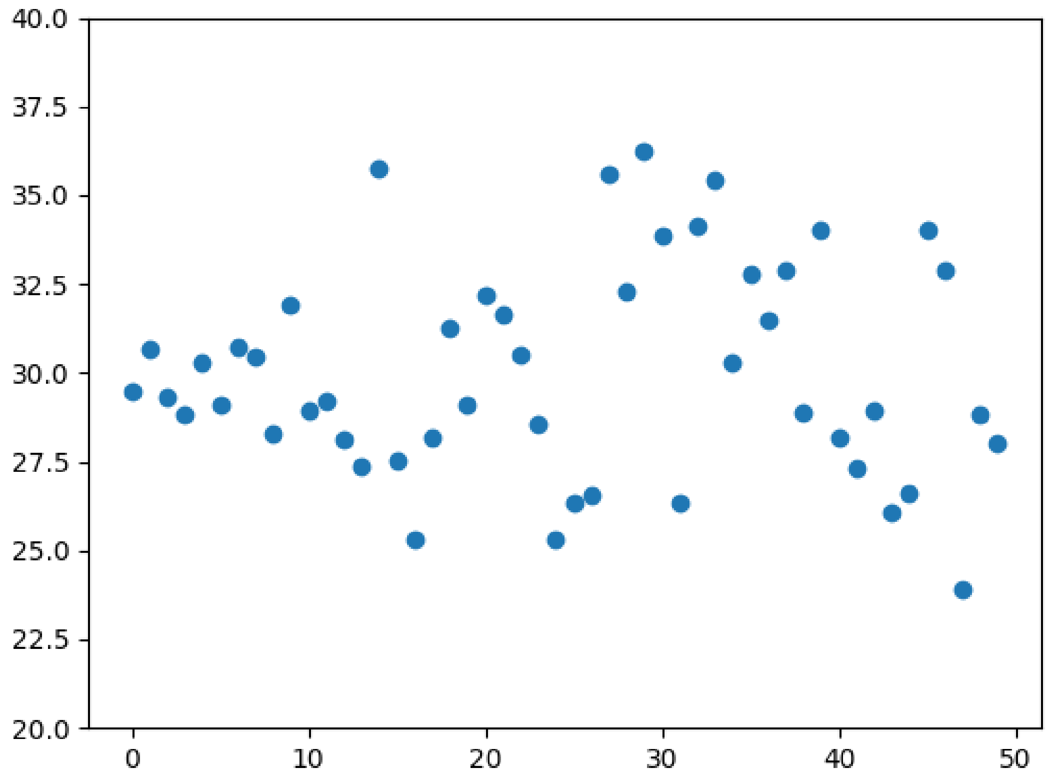

Figure 8: For k=50 in option b

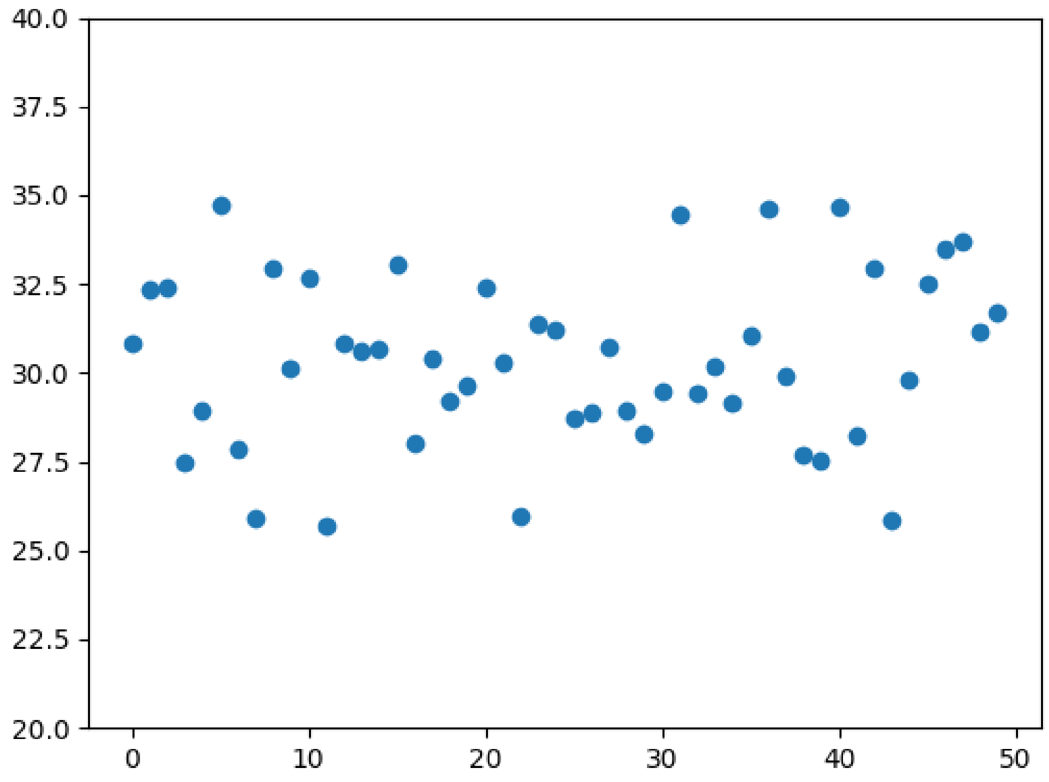

Figure 9: For k=100 in option b

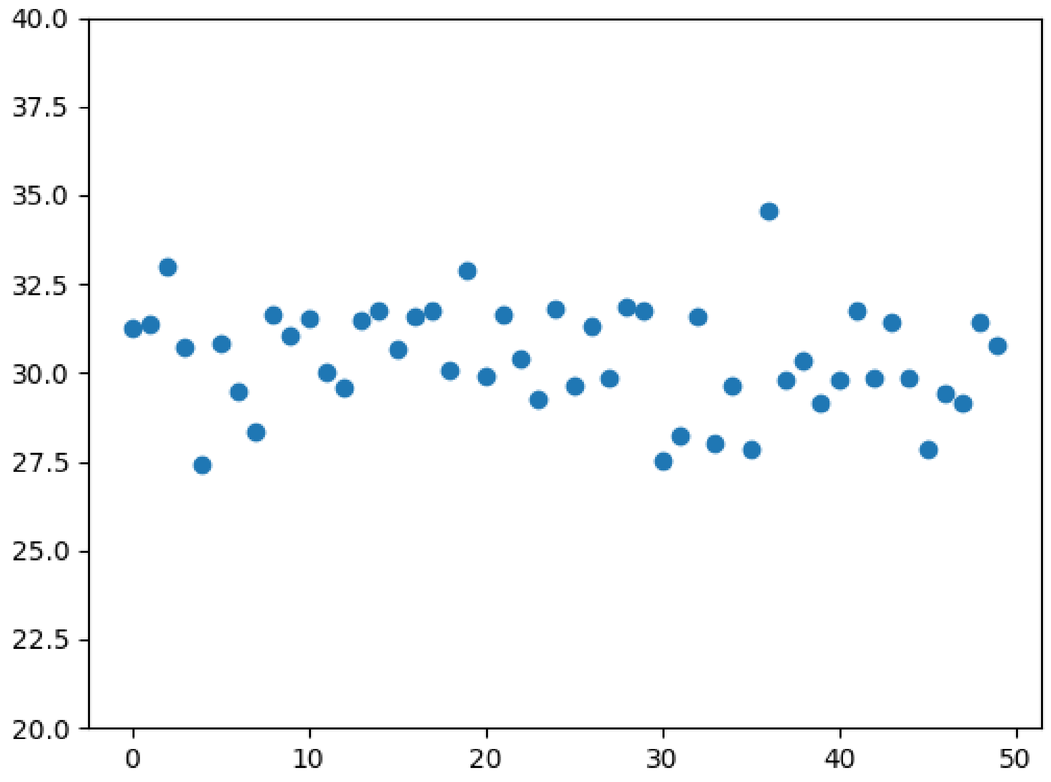

Figure 10: For k=200 in option b

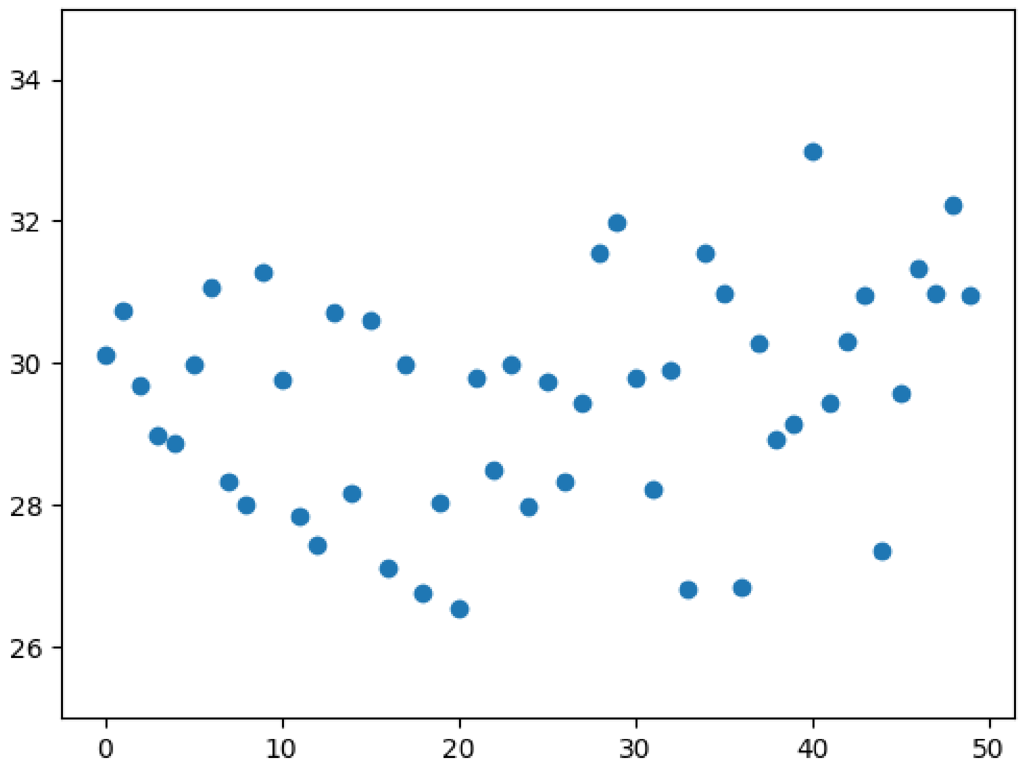

Figure 11: For k=10 in option c

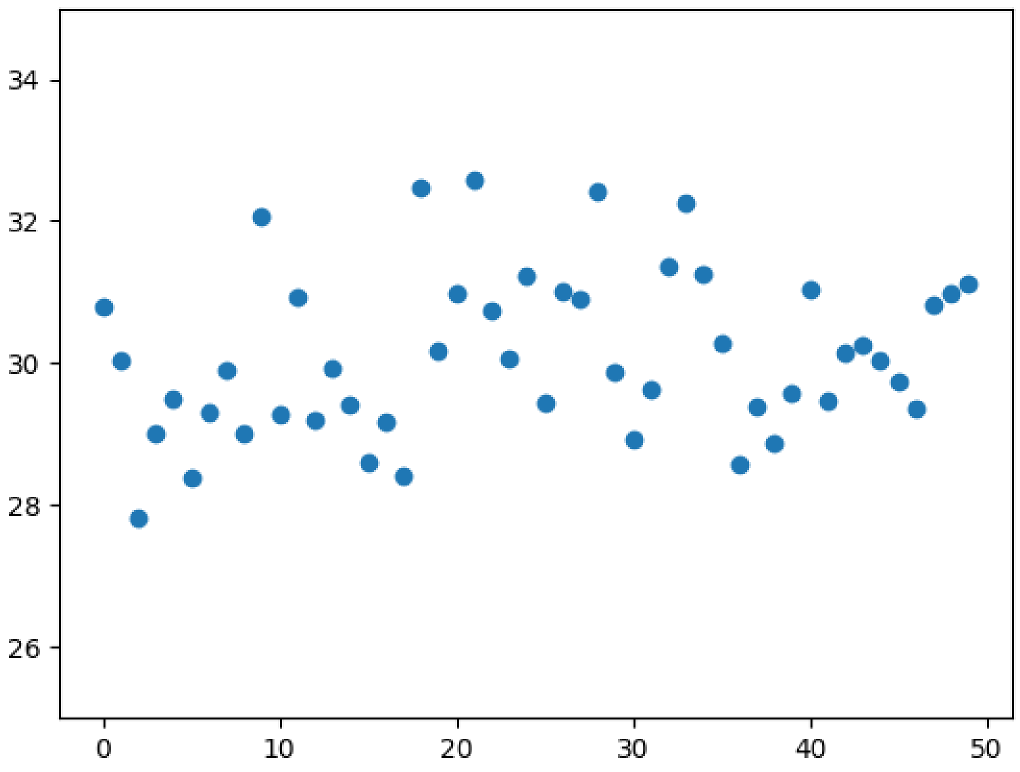

Figure 12: For k=20 in option c

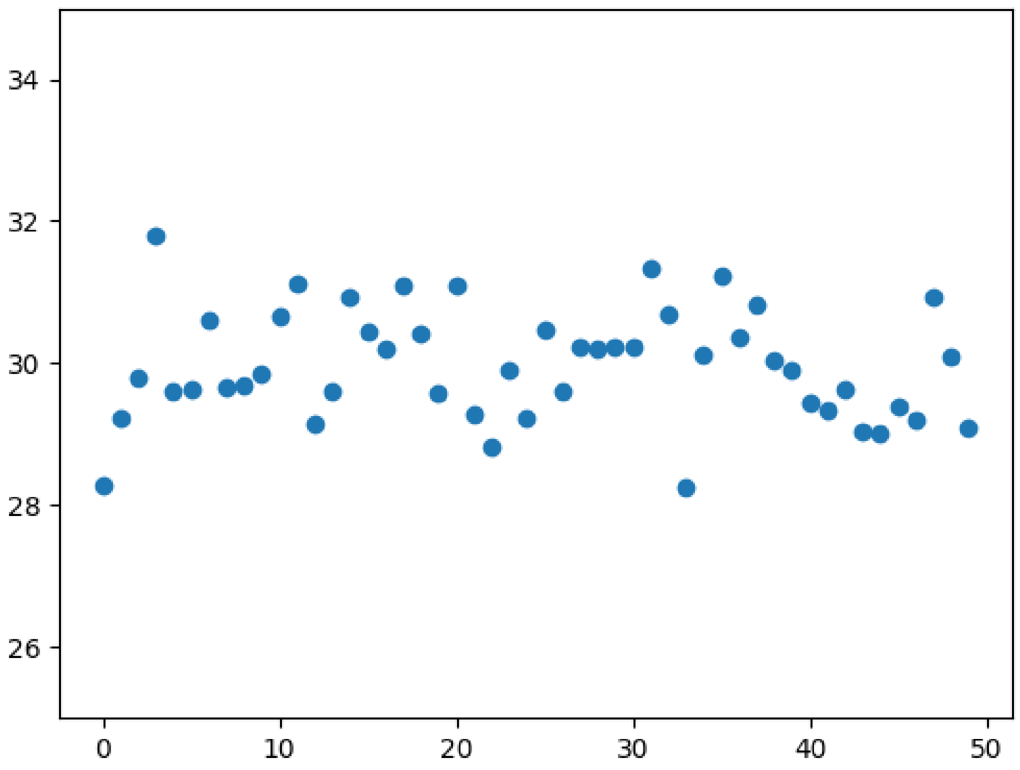

Figure 13: For k=50 in option c

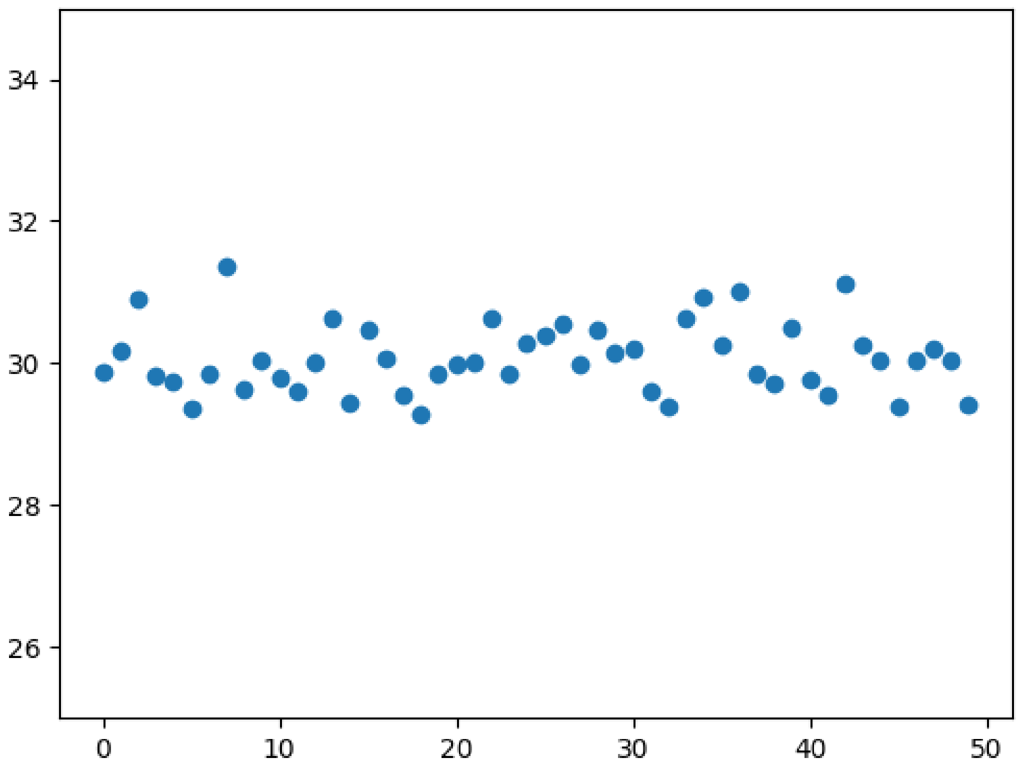

Figure 14: For k=100 in option c

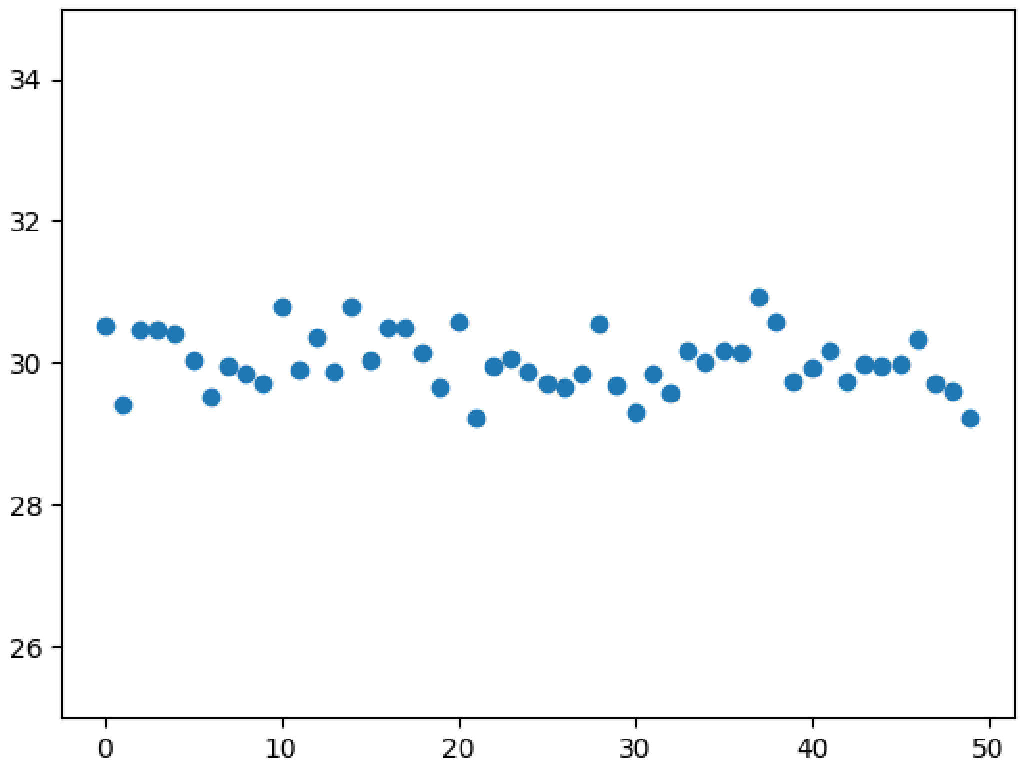

Figure 15: For k=200 in option c

1.2 part (b)

(i)

My guess for average value of data is 30 and for standard deviation is 5.
Actual Values: Average is 30.04963629765155
Standerd Deviation is 4.936015157083942

(ii) part A

From scatter plots, we can see that deviation from average value is more in case a and b. while in case c, we get less deviation means there are high chances that our experiment will give outcomes close to average value.

(ii) part B

I will use K as max as possible. Here in this question, I will use k=200. Because from scatter plots we can see that standard deviation is least for k=200. So I used "standerd deviation" as quantitative measure for sureness here.Lesser the standerd deviation, I will be more sure that data will be more close to actual average value.

2 Question 2

Since Kohli always chose heads, he would have believed probability of Head is greater than 0.5. Suppose he assumed probability of head to be 0.55.

Kohli believes that his opponent do not have powers of shakuni means he can't manipulate result of coin. Also kohli believes that coin is not sholay coin which was double head coin which gave head all times.

code for the question 2:

```
import random
import matplotlib.pyplot as plt
import numpy as np
import statistics
from statistics import mean
"""
reading data from all 4 files into lines arrays
"""
text_file = open("hw1b1.txt", "r")
lines1= text_file.readlines()
text_file.close()

text_file = open("hw1b2.txt", "r")
lines2= text_file.readlines()
text_file.close()
```

```
text_file = open("hw1b3.txt", "r")
lines3= text_file.readlines()
text_file.close()

text_file = open("hw1b4.txt", "r")
lines4= text_file.readlines()
text_file.close()
"""
Since lines arrays have elements of data types string,
we first converted them to int data type in new arrays num
"""
num1 = []
for i in range(len(lines1)):
    t = int(lines1[i])
    num1.append(t)

num2 = []
for i in range(len(lines2)):
    t = int(lines2[i])
    num2.append(t)

num3 = []
for i in range(len(lines3)):
    t = int(lines3[i])
    num3.append(t)

num4 = []
for i in range(len(lines4)):
    t = int(lines4[i])
    num4.append(t)

xh=0
xt=0
ph=0    #ph is probability of head
nh=0     #nh is no of heads appeared till now

for i in range(100):
    if num1[i]==1:
        nh=nh+1
    ph=nh/(i+1)
    if i>5 and ph>0.7 :
        xh+=1
    if i>5 and ph<0.4 :
        xt+=1
    if xh==10 :
         print("Coin may be biased towards head")
         print(i+1)
    if xt==10 :
        print("Coin may be biased towards tails")
        print(i+1)
    if xh==15 :
         print("Coin is biased towards head")
         print(i+1)
         break
    if xt==15 :
        print("Coin is biased towards tails")
        print(i+1)
        break
    if i==99:
         print("coin is not biased")

"""
We will skip first 5 tosses because initially value of probability
may change significantly. But after some experiments, it will become steady.
Then we can make decisions with more surity.
"""
```

(a)He will start doubting when probability of head differs by 0.15 with his assumed value 0.55 more than 10 times.

(b) He will be sure when probability will differ by 0.15 more than 15 times. Here if probability goes above 0.7 more than 15 times then coin is biased towards heads. If it goes below 0.4 more than 15 times then it is biased towards tails.

For data given in hw1b1.txt ; kohli will start doubting after 21 tosses and he will be sure after 27 tosses that coin is biased towards heads.

(c) For hw1b2.txt-hw1b4.txt use arrays num2, num3 and num4 instead of num1 in for loop of the code.

For hw1b2.tx, kohli will start doubting after 16 tosses and he will be sure after 21 tosses that coin is biased towards tails.

For hw1b3.tx, kohli will find that coin is not biased.

For hw1b4.tx, kohli will start doubting after 16 tosses and he will be sure after 21 tosses that coin is biased towards tails.

(d)As more and more times coin is tossed, probability of head(which is ratio of no of times head appears and total no of tosses)gets more and more close to its actual value(probability of head in each toss). So if probability of heads differs significantly from his assumed value more and more times then kohli will get more and more sure that coin is biased.

3 Question 3

To minimize the root mean square error, SSE= $\sum_{i=1}^{n}(y_i - ax_i - b)^2$:

$$\frac{\partial(SSE)}{\partial a} = 0 \implies a = \frac{\sum_i x_i y_i - n\bar{x}\bar{y}}{\sum_i x_i^2 - n\bar{x}^2}$$

Similarly for b we get,

$$b = \bar{y} - a\bar{x}$$

On substituting the values, we get the corresponding values of a and b. This concept has been used to build the model for hw1c1.txt.
This is the code for the model:

```python
import numpy as np
import matplotlib.pyplot as plt
import math as m
w=[]
h=[]
d1=open("hw1c1.txt",'r')
for line in d1.readlines():
    h.append(float(line[1:line.index(',')]))
    w.append(float(line[line.index(',')+1:line.index(')')
        ]))
x_bar=np.mean(h)
y_bar=np.mean(w)
pro=[]
sq=[]
def sse(h,w):
    for i in range(0,50):
        pro.append(h[i]*w[i])
        sq.append(h[i]**2)
    a=(np.sum(pro)-50*x_bar*y_bar)/(np.sum(sq)-50*(x_bar
        **2))
    b=y_bar-a*x_bar
   return a,b
a,b=sse(h,w)
reg=[a*h[i]-b for i in range(0,50)]
plt.plot(h,reg,'r')
plt.plot(h,w,'bo',markersize=3)
plt.xlabel('Heights')
plt.ylabel('Weights')
error=[(-reg[i]+w[i]) for i in range(0,50)]
```

a) Using the model to predict the expected value and comparing with the value observed generates the error. This has been plotted and the mean and standard deviation has been found out.

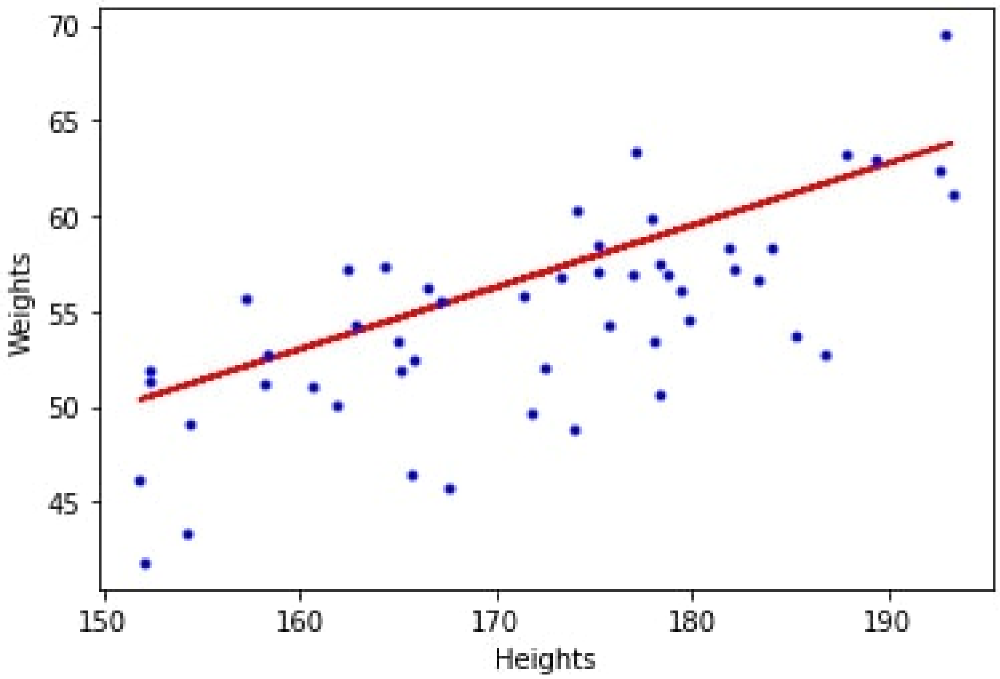

Figure 16: Model generated for fitting the data in hw1c1.txt.

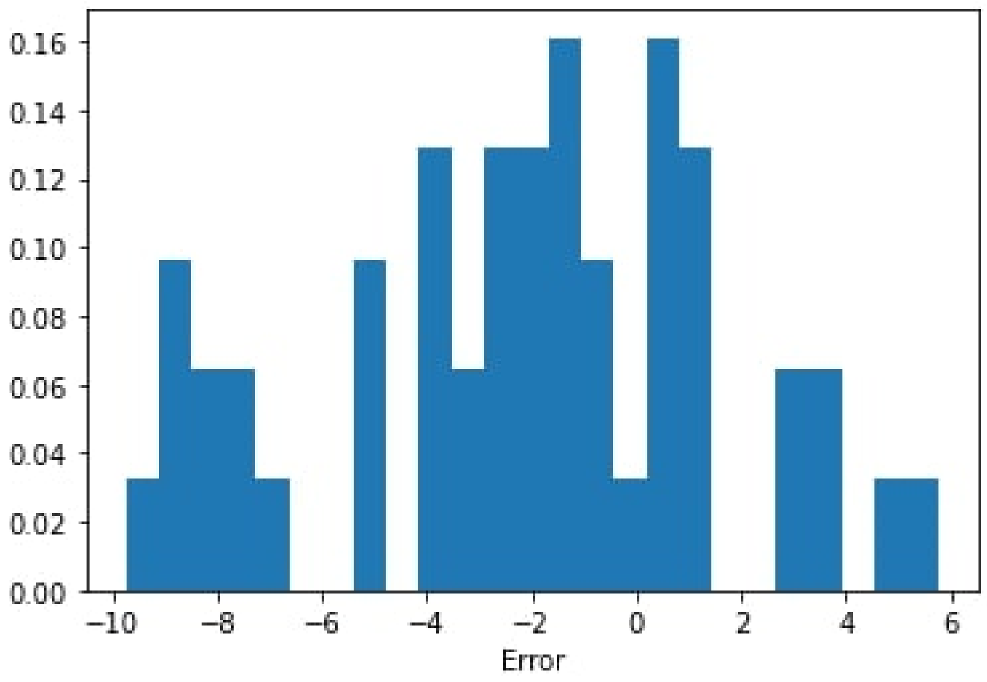

Figure 17: Error plot

```python
import seaborn as sns
plt.hist(error,bins=25,density=True)
plt.xlabel('Error')
np.mean(error)
np.std(error, ddof=1)
```

Here mean error= -2.188831586969617 and standard deviation= 3.811460298166977
The plot here is approximately like a Gaussian function.

b) For the model to be justified, the standard deviation of the three errors pertaining to the three different heights should also be close to the standard deviation of the error found earlier.

```
d2=open("hw1c2.txt",'r')
wt=[]
w_155=[]
w_165=[]
w_175=[]
i=0
l=d2.readlines()
w_155=[float(l[i]) for i in range(1,26)]
w_165=[float(l[i]) for i in range(27,52)]
w_175=[float(l[i]) for i in range(53,78)]
ycap_155=a*155-b
ycap_165=a*165-b
ycap_175=a*175-b
er1=[w_155[i]-ycap_155 for i in range(0,25)]
er2=[w_165[i]-ycap_165 for i in range(0,25)]
er3=[w_175[i]-ycap_175 for i in range(0,25)]
plt.hist(er1,bins=10, density=True)
plt.xlabel('Error: 155 cm')
np.std(er1, ddof=1)
np.mean(er1)
plt.hist(er2,bins=10, density=True)
plt.xlabel('Error: 165 cm')
np.std(er2,ddof=1)
np.mean(er2)
plt.hist(er3,bins=10, density=True)
plt.xlabel('Error: 175 cm')
np.std(er3,ddof=1)
np.mean(er3)
```

For 155 cm: Mean= -1.5308223261406826 and Standard deviation= 4.006812857223591
For 165 cm: Mean= -0.8040571918109896 and Standard deviation= 3.836492804194251
For 175 cm: Mean= -2.2641732404374117 and Standard deviation= 4.441680332371082

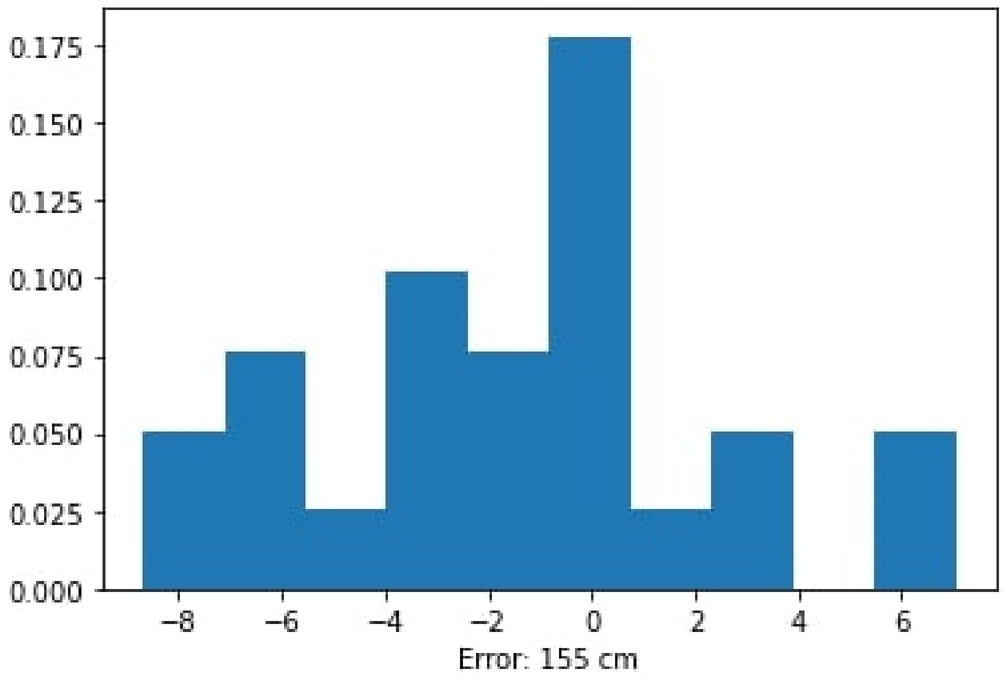

Figure 18: For 155 cm

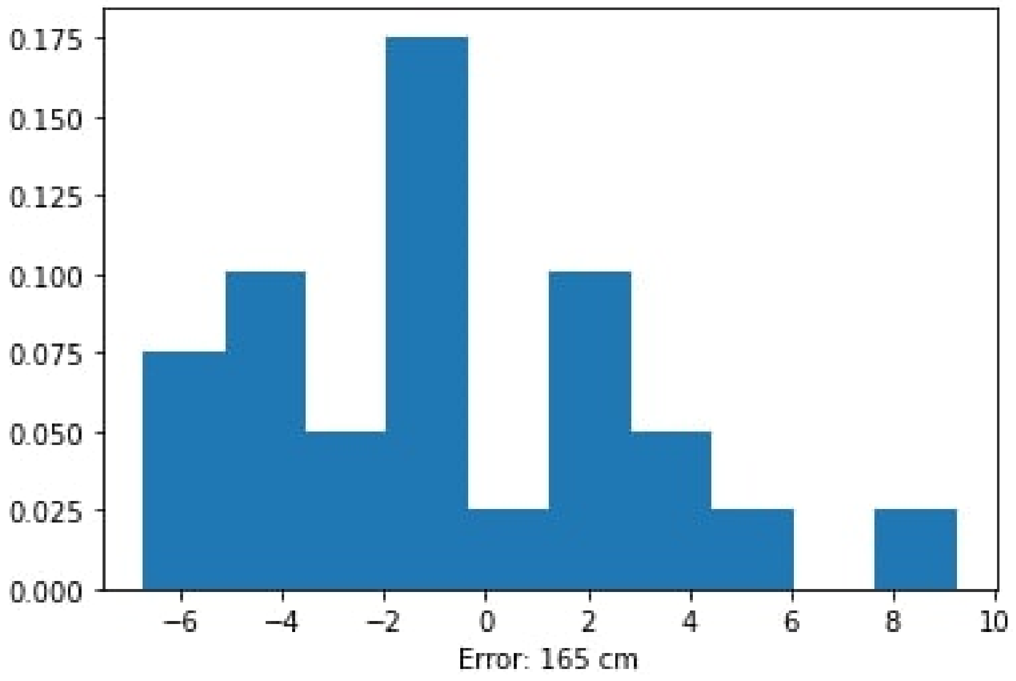

Figure 19: For 165 cm

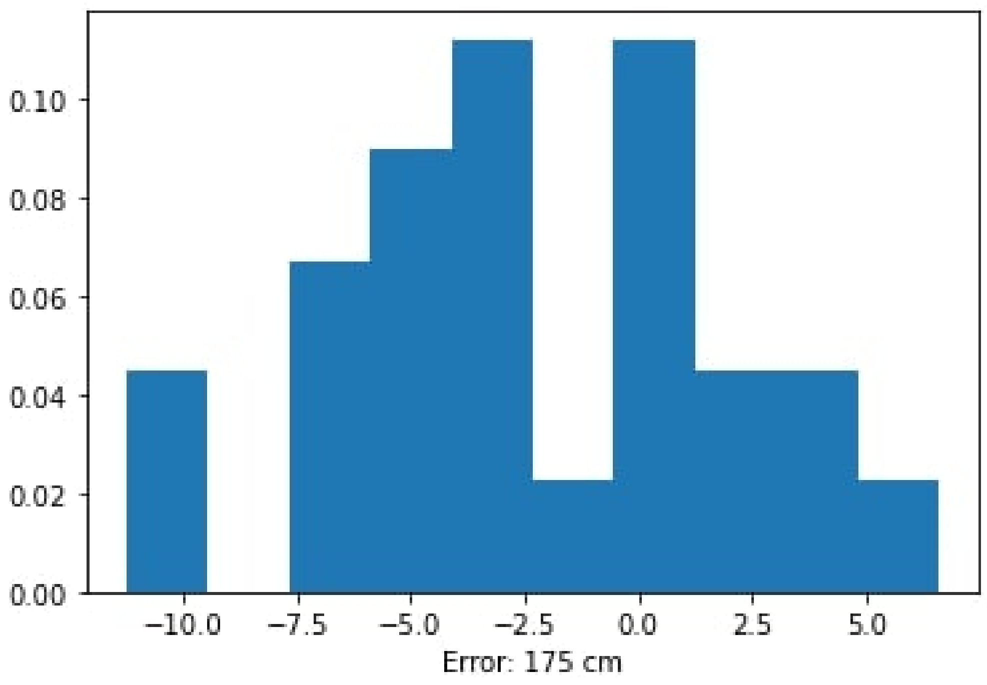

Figure 20: For 175 cm