Lab 1

Aditya Kharosekar amk3587

Eyeunwana Nwoko ein79

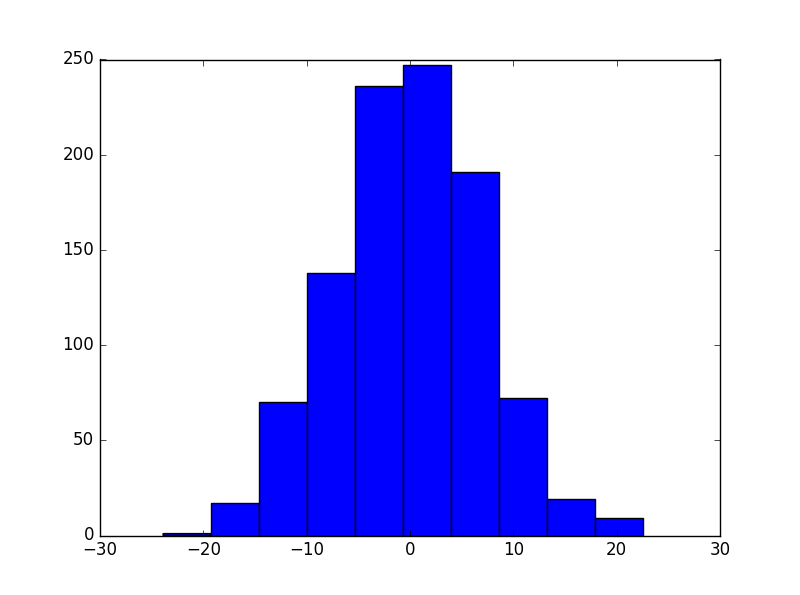
# Problem 1

## Part a

The sum of two Gaussian distributions is also a Gaussian.

## Part b

On one run of the script, the mean was 0.03069 and the variance was 7.0128.



Code –

import numpy as np

import matplotlib.pyplot as plt

first = np.random.normal(-10.0, 5.0, 1000)

second = np.random.normal(10.0, 5.0, 1000)

mean = np.mean(first+second)

variance = np.std(first+second)

print(mean, variance)

plt.hist(first+second)

plt.show()

# Problem 2

import numpy as np

import matplotlib.pyplot as plt

p = .5

first = 5

second = 50

third = 250

sum1\_array = []

sum2\_array = []

sum3\_array = []

for i in range(1000):

sum1 = 0

sum2 = 0

sum3 = 0

rv = np.random.binomial(1, p, first)

for num in rv:

num = (num \* 2) - 1

sum1 += num

rv = np.random.binomial(1, p, second)

for num in rv:

num = (num \* 2) - 1

sum2 += num

rv = np.random.binomial(1, p, third)

for num in rv:

num = (num \* 2) - 1

sum3 += num

sum1\_array.append(sum1/float(first))

sum2\_array.append(sum2/float(second))

sum3\_array.append(sum3/float(third))

bins = np.linspace(-1, 1, 32)

plt.hist(sum1\_array, bins=bins)

plt.show()

bins = np.linspace(-1, 1, second)

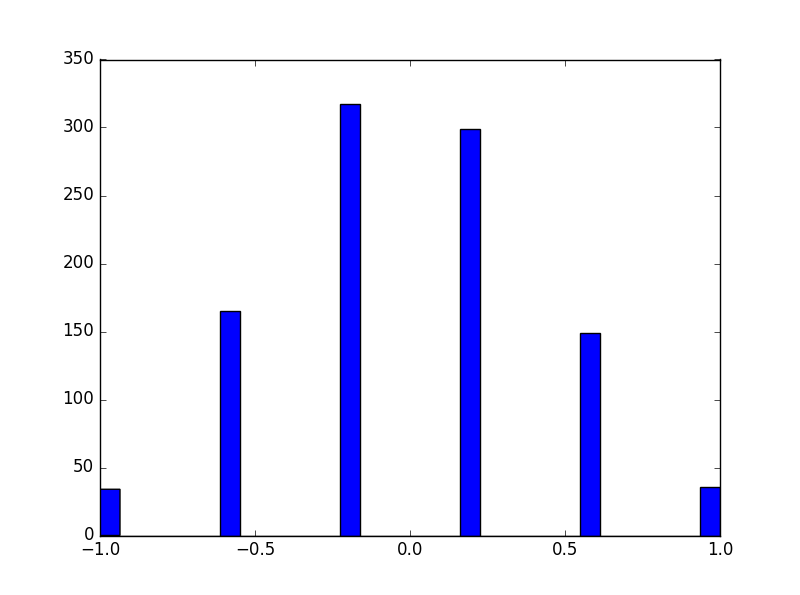
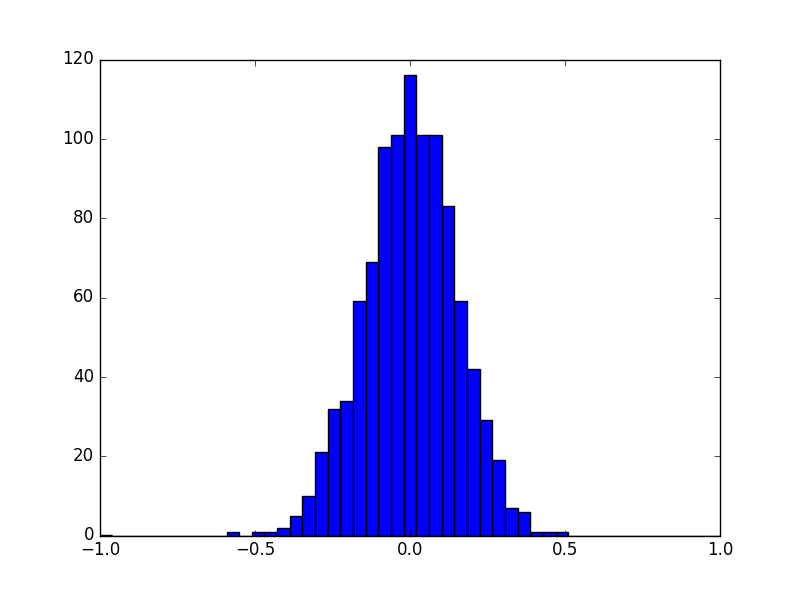
plt.hist(sum2\_array, bins=bins)

plt.show()

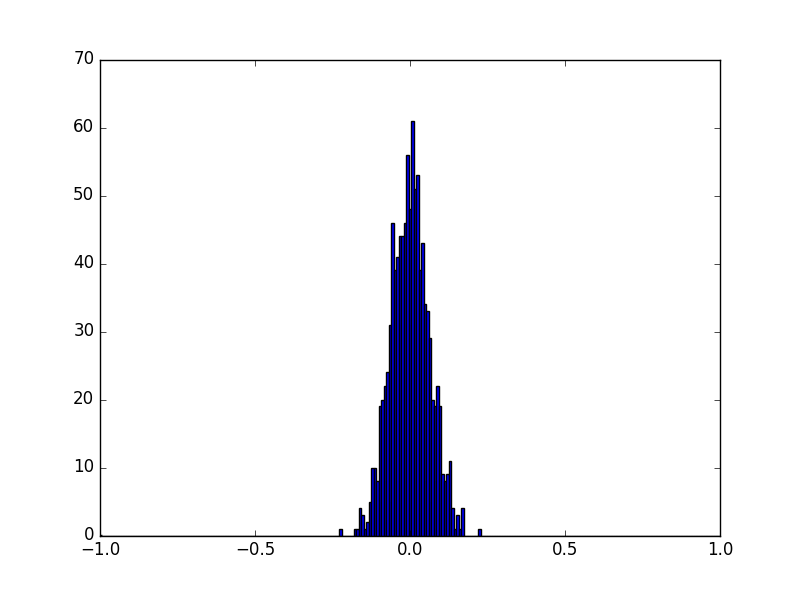
bins = np.linspace(-1, 1, third)

plt.hist(sum3\_array, bins=bins)

plt.show()

With n=5

With n=50



With n=250

# Problem 3

import numpy as np

rv = np.random.normal(loc=0.0, scale = 5.0, size=25000)

mean = np.sum(rv)/25000

sum = 0.0

for num in range(0, 25000):

difference = rv[num]-mean

sum = sum + (difference \*\*2)

stddev = (sum/25000) \*\* (0.5) #Taking the square root

print(mean)

print(stddev)

For one run of the script, the mean was -0.0337 and the variance was 5.1055

# Problem 4

import numpy as np

mean = np.array([-5, 5])

cov = np.array([[20, 0.8], [0.8, 30]])

rv = np.random.multivariate\_normal(mean, cov, 10000)

number = 10000

sum = 0.0

for num in range(0, number):

sum = sum + rv[num]

mean1 = sum/number

meanX = mean1[0]

meanY = mean1[1]

x = []

y = []

sumX = 0.0

sumY = 0.0

differenceX = 0.0

differenceY = 0.0

covSum = 0.0

for i in range(0, number):

x.append(rv[i][0])

y.append(rv[i][1])

for i in range(0, number):

differenceX = x[i]-meanX

sumX = sumX + (differenceX\*\*2)

differenceY = y[i] - meanY

sumY = sumY + (differenceY\*\*2)

covSum = covSum + (differenceX \*differenceY)

varX = sumX/number

varY = sumY/number

stddevX = varX \*\* 0.5

stddevY = varY \*\* 0.5

covariance = covSum/number

correlation = covariance/(stddevX \* stddevY)

print(mean1)

print("Covariance Matrix: ")

print(varX, (correlation \* stddevX \* stddevY), (correlation \* stddevX \* stddevY), varY)

On one run of the script, the mean was [-4.95, 4.98] and the correlation matrix was [[19.82, 0.76], [0.76, 29.23]]

# Problem 5

## Part a

There are 452 patients and 279 features for each patient.

import pandas as pd

df = pd.read\_csv("PatientData.csv")

numPatients = len(df.index) + 1

#There are 452 patients

numFeatures = len(df.columns)

#There are 279 features

## Part b

The first feature looks like the patient’s age. The second feature looks like a flag to show patient gender (For example: 1 if female, 0 if male). The third feature looks the patient’s height in centimeters. The fourth feature looks like the patient’s weight in kilograms.

## Part c

Yes, there are missing values in various columns. Df.mean() creates a Pandas series in which each element is the mean of the corresponding feature’s values.

df.fillna(df.mean())

## Part d

I will use a binary decision tree to calculate the importance of each feature.

## Part e

from sklearn.ensemble import ExtraTreesClassifier

from sklearn.feature\_selection import SelectFromModel

import pandas as pd

patient\_data = pd.read\_csv('PatientData.csv', header=None, na\_values='?')

patient\_data = patient\_data.fillna(patient\_data.mean())

clf = ExtraTreesClassifier()

clf = clf.fit(patient\_data.iloc[:, :-1], patient\_data.iloc[:, -1 ])

imp = list(clf.feature\_importances\_)

impfeatures = []

impfeatures.add(imp.index(max(imp)))

imp.remove(max(imp))

impfeatures.add(imp.index(max(imp)))

imp.remove(max(imp))

impfeatures.add(imp.index(max(imp)))

imp.remove(max(imp))

print("The important features are : ", imp)

The three most important features were – 92, 90, 147.

# Written Question

