

**ANL252**

**Python for Data Analytics**

# **Tutor Marked Assignment**

**July 2021 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Brandon Tan Xuan Ming** | **Y1970855** |

**Instructor’s Name: Mr. KUMAR MUNISH**

**Submission Date: 15/08/2021**

1a)

﻿# Importing math module

import math

1b)

﻿# Program to ask the user to enter the mean and variance of the distribution

mean = 0

input\_mean = True

while input\_mean:

try:

user\_mean = input("Please insert mean, where mean = (-∞,+∞):")

if user\_mean == '':

mean = 0

print('No input detected, mean is set to 0')

input\_mean = False

else:

mean = float(user\_mean)

input\_mean = False

except ValueError:

print('Error, invalid mean')

continue

variance = 1

input\_var = True

while input\_var:

try:

user\_var = input("Please insert variance, where variance > 0:")

if user\_var == '':

variance = 1

print('No input detected, variance is set to 1')

input\_var = False

else:

variance = float(user\_var)

if variance > 0:

input\_var = False

else:

print('Error, invalid variance')

continue

except ValueError:

print('Error, invalid variance')

continue

print('User Input: Mean =', mean,',',

'Variance =', variance)

Program output for (b) if User input ENTER without providing details for both mean and variance:

Text

Description automatically generated

1c)

﻿# Input screen for the user to enter the value of X

X = 0

input\_X = True

while input\_X:

try:

user\_X = input("Please insert X, where X = (-∞,+∞):")

if user\_X == '':

print('No input detected')

continue

else:

X = float(user\_X)

input\_X = False

except ValueError:

print('Error, invalid X')

continue

print('User Input: X =', X)

Program output for (c) if User input non numeric for X:

Text

Description automatically generated

1d)

﻿# Using the formula as defined in the question

pdf1 = 1/(math.sqrt(2\*math.pi\*variance))

pdf2 = math.exp(-((X-mean)\*\*2)/(2\*variance))

pdf = pdf1\*pdf2

print('Probability that X =', X, ':', pdf)

Program output for (d) to compute the corresponding probability density fX(x) based on the user inputs in (b) and (c):



1e)

Use formatted printing to display the result of (d) to the user:

Text

Description automatically generated

1f)

﻿# specify the function CDF such that it will return the probability that x <= X when keyed in.

# refer to standard normal distribution. where CDF can be written as

# 0.5\*{1 + erf[(X-mean)/sqrt(2\*variance)]}

def CDF(X):

g = math.erf((X-mean)/math.sqrt(variance\*2.0))

f = 0.5\*(1.0 + g)

return f

print('Probability that X <=', X, ':', CDF(X))

# specify the function PDF to solve for the probability of a normally distributed random variable

# again, as defined by the question

def PDF(X,mean,variance):

B = (X-mean)/variance

A = (1/(math.sqrt(2\*math.pi\*variance)))\*(math.exp(-(B\*B)/2))

return A

print('Probability that X =', X, ':', PDF(X,mean,variance))

# Check the results for k = 0, 1.64, 1.96, and mean = 0, variance = 1

# For this, we can re-run this entire cell to produce the answers.

# Based on our program, when k = 0, P(X = 0): 0.3989 & P(X <= 0): 0.50

# when k = 1.64, P(X = 1.64): 0.1040 & P(X <= 0): 0.9495

# when k = 1.96, P(X = 1.96): 0.0584 & P(X <= 0): 0.9750

# ------------------------------------------ Checking answers for part F -----------------------------------------------

# We can check the results of the above manually

# When we say standard, we use the formula for when mean = 0, variance = 1

# This is defined in the documentation for python math package, under erf function

def standard\_CDF(Y):

t = math.erf(Y/math.sqrt(2.0))

return((1.0 + t)/2.0)

Y = [0, 1.64, 1.96]

for y in Y:

print(standard\_CDF(y))

# equation as per defined in the question

def standard\_PDF(Y):

u = 1/(math.sqrt(2\*math.pi\*1))

v = math.exp(-((Y-0)\*\*2)/(2\*1))

return u\*v

for y in Y:

print(standard\_PDF(y))

Checking (f) for k = 0, k = 1.64, k = 1.96. Where CDF = P(X = 0), and PDF = P(X<=0), mean = 0 and variance = 1:

Text

Description automatically generated

1g)

As defined by the cumulative distribution function:

Text, letter

Description automatically generated

We will need to rewrite the equation into :

﻿0.5\*{1 + erf[(X-mean)/sqrt(2\*variance)]}

And to further simplify the equation, we split the equation into 2 portion where I first define the back portion of erf((X-mean)/math.sqrt(variance\*2.0)):

﻿def CDF(X):

g = math.erf((X-mean)/math.sqrt(variance\*2.0))

f = 0.5\*(1.0 + g)

return f

Therefore, we can define that ﻿Probability that X <= User (x)

While defining the Probability Density function from the question:

Diagram

Description automatically generated with low confidence

I also simplified the equation to the following by splitting off the second portion for (X-mean)/variance) before forming the whole equation:

﻿def PDF(X,mean,variance):

B = (X-mean)/variance

A = (1/(math.sqrt(2\*math.pi\*variance)))\*(math.exp(-(B\*B)/2))

return A

We can also find the probability that X = User (x)

1h)

﻿m = 0 # Mean

var = 1 # Variance

# Creating the list 'X' such that it is -5 to 5. Intervals of 0.1 such that we can use the step width of 0.5

X\_list = [0.1\*x for x in range(-50,51)]

# Creating dictionary with the results, 'prob', which is just calculating pdf for each x in 'X'

X\_dict = {}

for x in X\_list:

prob = (1/(math.sqrt(2\*math.pi\*var)))\*(math.exp(-((x-m)\*\*2)/(2\*var)))

X\_dict[x] = prob

# myX is the selection of X's in range -2 to 2 where the step width is 0.5

myX = [0.1\*x for x in range(-20,21,5)]

# Printing the results

for x in myX:

print('x =', x, ',', 'p(x) =', X\_dict[x])

probabilities (with the corresponding x) of those x’s between -2 and 2 with a step width of 0.5 from the dictionary onto the screen:

Text

Description automatically generated