NORTHWESTERN UNIVERSITY | SCHOOL OF PROFESSIONAL STUDIES

Fall 2017 Final Exam

PREDICT 400: Math for Modelers

Points possible: 100

Description: The final exam will cover topics from sessions 1-9.

Resources: The exam is completely open book. You may use course textbooks, materials provided on Canvas, graphing calculators (such as TI 83 or 84); *but any more advanced calculators, Excel Solver, Web calculators, Web-graphic calculators, or simplex method calculators are not allowed. Programming languages other than Python are also not permitted*.

For questions that require calculations, all calculations should be shown, not just the final answer. This will allow for partial credit for those answers that might be set up correctly but have calculation errors. If variables are introduced, whether in a program or in a mathematical solution, you will need to explicitly state what the variables represent and clearly state the results based on the variable designations.

For questions that specifically require Python, your code and output should be included to constitute work shown along with solutions. For questions that require graphs, only use Python. Python can be used for all questions unless instructed otherwise. For any problem the uses Python, your code and output must be provided.

Restrictions: All answers are to be your work only. You are not to receive assistance from any other person.

To complete the exam:

1. Answer all questions on the exam thoroughly. Create a single Microsoft Word document, including the question number, the question, your typed answer, and graphs if required. You may use Word's equation editor to complete your answers.

2. Once you have completed your exam, save the file with a meaningful filename such as "Final" followed by your last name and return to the exam item where you downloaded the exam PDF, click View/Complete Assignment, and submit your document.

1. A local copy center needs to buy white paper and yellow paper. They can buy from three suppliers. Supplier 1 sells a package of 20 reams of white and 10 reams of yellow for $60. Supplier 2 sells a package of 10 reams of white and 10 reams of yellow for $40. Supplier 3 sells a package of 10 reams of white and 20 reams of yellow for $50. The copy center needs 350 reams of white and 400 reams of yellow. **Using Python**, determine (1) how many packages they should buy from each supplier in order to minimize cost and (2) the minimum cost.

```
from pulp import *


class LinearProgramming():

    def __init__(self):
        self.white_reams = LpVariable("white_reams", 350, 99999)     # White reams
required
        self.yellow_reams = LpVariable("yellow_reams", 400, 99999)   # Yellow reams
required

        self.package1 = LpVariable("package1", 0, 99999)  # Packages from Supplier One
        self.package2 = LpVariable("package2", 0, 99999)  # Packages from Supplier Two
        self.package3 = LpVariable("package3", 0, 99999)  # Packages from Supplier
Three

    # LP problem is defined by this function and is of the minimization category
    def define_equation(self):
        self.lp_eq = LpProblem("problem", LpMinimize)

    # This function defines the constraints on the equation
    def define_constraints(self):
        package_constraints_1 = self.package1 * 20
        package_constraints_2 = self.package2 * 10
        package_constraints_3 = self.package3 * 10
        self.lp_eq += package_constraints_1 + package_constraints_2 +
package_constraints_3 == self.white_reams
        package1_constraints_2 = self.package1 * 10
        package2_constraints_2 = self.package2 * 10
        package3_constraints_2 = self.package3 * 20
        self.lp_eq += package1_constraints_2 + package2_constraints_2 +
package3_constraints_2 == self.yellow_reams

    # Objective function for the LP is defined here
    def define_objective_function(self):
        package1_objective = self.package1 * 60
        package2_objective = self.package2 * 40
        package3_objective = self.package3 * 50
        self.lp_eq += package1_objective + package2_objective + package3_objective

    # Problem is initialized and defined in the functions above.
    # This function returns the package quantity that must be purchased from each
supplier.
    def problem_solver(self):

self.lp_eq.solve(solver=GLPK("C:\\Users\\User\\Anaconda3\\Library\\bin\\glpsol.exe"))
        self.package1 = round(value(self.package1), 0)  # Rounds off the value of
package 1
        self.package2 = round(value(self.package2), 0)  # Rounds off the value of
```

```
package 2
        self.package3 = round(value(self.package3), 0)  # Rounds off the value of
package 3

    # This function prints out the solution of the LP problem defined and solved
above.
    def print_solution(self):
        print("Packages that should be bought from Supplier 1 in order to minimize
cost:", self.package1)
        print("Packages that should be bought from Supplier 2 in order to minimize
cost:", self.package2)
        print("Packages that should be bought from Supplier 3 in order to minimize
cost:", self.package3)

        min_total_cost = self.package1 * 60 + self.package2 * 40 + self.package3 * 50
        print("The Minimum Cost is:", min_total_cost)



def main():
    app = LinearProgramming()
    app.define_equation()
    app.define_constraints()
    app.define_objective_function()
    app.problem_solver()
    app.print_solution()


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

Packages that should be bought from Supplier 1 in order to minimize cost: 10
Packages that should be bought from Supplier 2 in order to minimize cost: 0
Packages that should be bought from Supplier 3 in order to minimize cost: 15

The Minimum Cost is: $1350.00

Supplier 1 sells x packages
Supplier 2 sells y packages
Supplier 3 sells z packages

so total cost $= 60x + 40y + 50z$
so we have following linear programming problem.
minimize 60x+40y+50z
subject to
$20x + 10y + 10z = 350$
$10x + 10y + 20z = 400$
$x \geq 0, y \geq 0, z \geq 0$

so subtract given equations
$x = z - 5$
$y = 45 - 3z$
so objective function becomes
$60z - 300 + 1800 - 120z + 50z$
$= 1500 - 10z$
since $x = z - 5$
$z \geq 5$
$45 - 3z \geq 0$
$z \leq 15$
and objective function will be minimum when z is maximum.
maximum value of z is 15.
so
$x = 10$
$y = 0$
$z = 15$
and minimum cost is 600+750 = 1350

2. A new test has been developed to detect a particular type of cancer. A medical researcher selects a random sample of 1,000 adults and finds (by other means) that 4% have this type of cancer. Each of the 1,000 adults is given the new test and it is found that the test indicates cancer in 99% of those who have it and in 1% of those who do not. Based on these results, what is the probability of a randomly chosen person having cancer given that the test indicates cancer? What is the probability of a person having cancer given that the test does not indicate cancer? Round the probabilities to four decimal places.

(a)Based on these results, what is the probability of a randomly chosen person having cancer given that the test indicates cancer?

Let

D = has cancer disease

P = tests positive

As by Bayes' Rule,

P (P) = P (D) P (P|D) + P (D') P (P|D') = 0.04*0.99 + (1-0.04)*0.01 = 0.0492

Hence,

P (D| P) = P(D) P(P|D)/P(P) = 0.04*0.99/0.0492 = 0.804878049

(b) What is the probability of a person having cancer given that the test does not indicate cancer?

P (D|P') = P(D) P(P'|D)/P(P') = 0.04*(1-0.99)/(1-0.0492) = 0.000420698

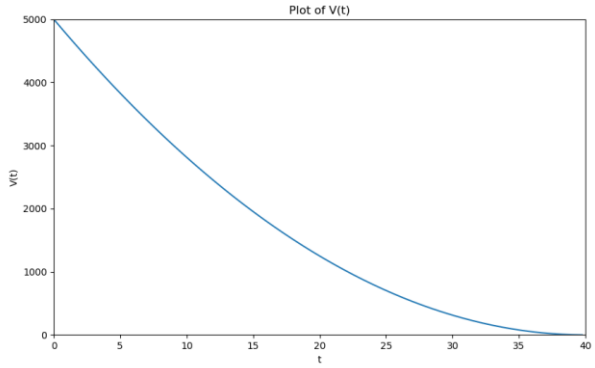
The probability of a randomly chosen person having cancer given that the test indicates cancer is = **0.809**

The probability of a person having cancer given that the test does not indicate cancer is = **0.0004**

3. If a tank holds 5000 gallons of water, which drains from the bottom of the tank in 40 minutes, then the volume of the water remaining in the tank after t minutes is given by $V(t) = 5000\left(1 - \frac{t}{40}\right)^2, 0 \leq t \leq 40$. **Using Python**, determine the rate at which water is draining from the tank. When will it be draining the fastest?

```
import numpy as np
import matplotlib.pyplot as plt
from sympy import *
```

```
class RateCalculator():

    # Initialize the parameters
    def __init__(self):
        self.qty_water = 5000   # Quantity of water in gallons
        self.drain_time = 40    # Water drain time in minutes

    # Calculates the volume of water remaining in the tank
    # after t minutes which is passed as the argument to the function
    def calc_drain_rate(self, t):
        self.y = self.qty_water * (1 - t/40)**2
        return self.y

    # Initializes the graph with axis labels and title and x & y limit.
    def init_graph(self):
        plt.figure(figsize=(10, 6))
        plt.title("Plot of V(t)")
        plt.ylabel("V(t)")
        plt.xlabel("t")
        plt.xlim(0, 40)
        plt.ylim(0, 5000)
        self.init_plot_points()  # Calculates the x and y points to be plotted on the
graph
        plt.plot(self.x_val, self.y_val)  # Plots the points on the graph
        plt.show()  # Displays the graph


    # Calculates the x & y points to be plotted on the graph
    def init_plot_points(self):
        self.x_val = np.arange(0, 40, 0.25)  # Points to be plotted on x-axis
        self.y_val = self.calc_drain_rate(self.x_val)  # Points to be plotted on the
y-axis


t = Symbol("t")
volume = 5000*(1 - t/40)**2
volume_of_drain1 = volume.diff(t)
print("d_1(V(t)):", volume_of_drain1)


t = Symbol("t")
volume_of_drain1 = 25*t/4 - 250
print("T(min) is 0: {}\nT(min) is 5: {}".format(volume_of_drain1.subs(t, 0),
volume_of_drain1.subs(t, 5)))
print("T(min) is 10: {}\nT(min) is 30: {}".format(volume_of_drain1.subs(t, 10),
volume_of_drain1.subs(t, 30)))
print("T(min) is 40: {}".format(volume_of_drain1.subs(t, 40)))


def main():
    app = RateCalculator()  # Create instance of the class
    app.init_graph()  # Calls the function which initializes the graph and displays
it.



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



The rate at which the volume changes is given by $\frac{dV}{dt}$. The rate at which it flows out is the exact opposite, i.e., $-\frac{dV}{dt}$.

Therefore, $\mathbf{v} - \frac{dV}{dt} = 250\left(1 - \frac{t}{40}\right) = 250 - \frac{25}{4}t$

Sl.No.	T(min)	$-\frac{dV}{dt}$
1.	0	250
2.	5	875/4=218.75
3.	10	375/2=187.5
4.	30	125
5.	40	0

The rate is a linear function; the graph is a line of negative slope, so it is constantly going down. The ratio is thus fastest at the very first (t=0) and slowest at the very end (t=40).

T(min) is 0: **-250**
T(min) is 5: **-875/4**
T(min) is 10: **-375/2**
T(min) is 30: **-125/2**
T(min) is 40: **0**

4. A rectangular container with a volume of 475 ft^3 is to be constructed with a square base and top. The cost per square foot for the bottom is \$0.20, for the top is \$0.10, and for the sides is \$0.015. Find the dimensions of the container that minimize the cost. Round to two decimal places.

Volume of rectangular container $= x^2h = 475$ .....(1)4

Where x is side of square base.

Total expenditure $(E) = 0.20x^2 + 0.10x^2 + 0.015 \times 4 \times xh$ .....(2)

From equation (1) and (2), we have

$$E = 0.20x^2 + 0.10x^2 + 0.015 \times 4 \times x \times \frac{475}{x^2}$$

$$= 0.30x^2 + \frac{28.5}{x}$$

For minimum, $\frac{dE}{dx} = 0$

$$\frac{dE}{dx} = 0.60x - \frac{28.5}{x^2} = 0$$

$$x^3 = 47.5$$

$$x \approx 3.62\,ft$$

And height (h) $= \frac{475}{13.116} \approx 36.22\,ft$

The dimensions of the container that minimize the cost is = **36.22ft**.

5. Assume the total revenue from the sale of x items is given by $R(x) = 27\ln(6x + 1)$ while the total cost to produce x items is $C(x) = x/7$. Find the approximate number of items that should be manufactured so that profit is maximized. Justify that the number of items you found gives you the maximum and state what the maximum profit is.

$$P(x) = Revenue - Cost$$
$$= R(x) - C(x)$$
$$= 27\ln(6x+1) - \frac{x}{7}$$

For maximum or minima;

$$\frac{dP(x)}{dx} = 0 \text{; thus, } \frac{dP(x)}{dx} = \frac{27\times 6}{(1+6x)} - \frac{1}{7} = 0$$

$$6x + 1 = 27 \times 42$$
$$x = 188.83 \approx 189$$

Approximate number of items is to be manufactured in order to ensure profit ≈ 8189

$$\text{Maximum profit} = 27\ln(1135) - \frac{189}{7} = \$162.928$$

6. For the following function, determine the domain, critical points, intervals where the function is increasing or decreasing, inflection points, intervals of concavity, intercepts, and asymptotes where applicable. Use this information and **Python to graph** the function.

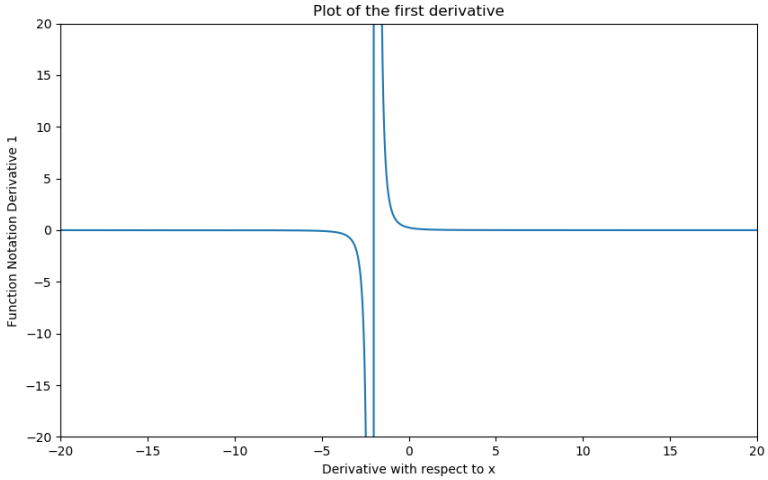
$$f(x) = -\frac{1}{(x+2)^2} + 4$$

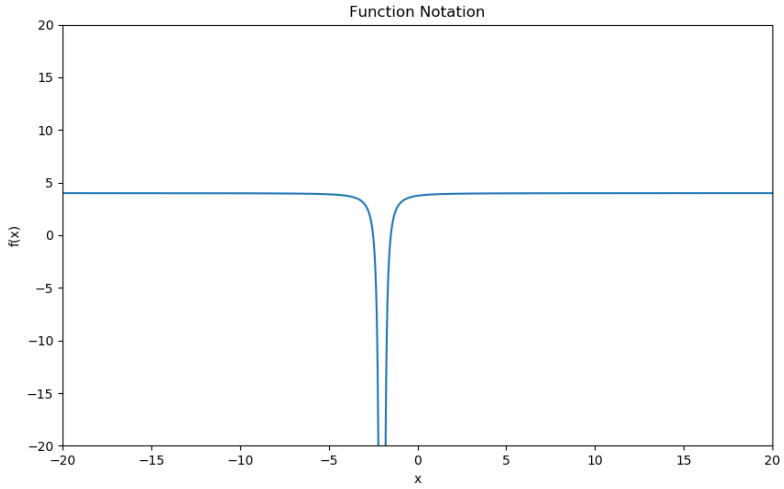
```
import numpy as np
import matplotlib.pyplot as plt
from sympy import *
import pandas as pd
```

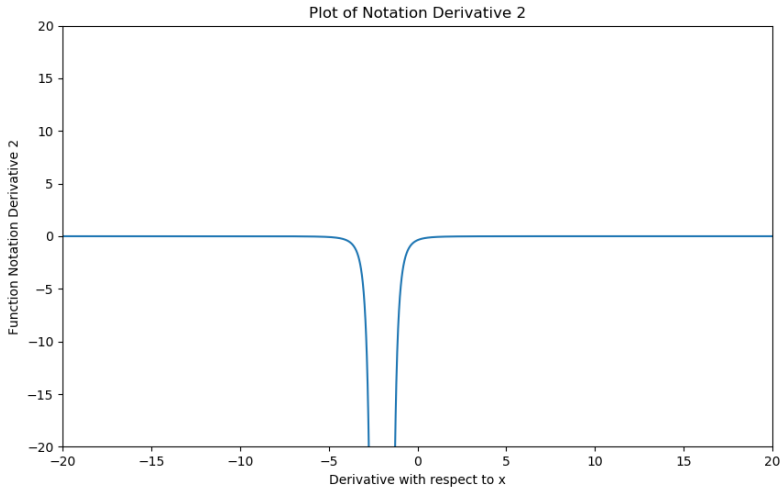
```
class FunctionGrapher():

    def __init__(self):
        self.x = Symbol("x")

    # The initial function
    def fn1(self, x):
        self.y = float(-1)/((x+2)**2)+4
        return self.y

    # The equation for the second derivative
    def fn2(self, x):
        self.y = float(2)/(x + 2)**3
        return self.y

    # The equation for the third derivative
    def f_d_3(self, x):
        self.y = -float(6)/(x+2)**4
        return self.y

    # Original function equation graph
    def init_graph_fn1(self):
        x = np.arange(-20, 20, 0.01)
        y = self.fn1(x)
        plt.figure(figsize=(10, 6))
        plt.plot(x, y)
        plt.xlim(-20, 20)
        plt.ylim(-20, 20)
        plt.title("Function Notation")
        plt.ylabel("f(x)")
        plt.xlabel("x")
        plt.show()

    # First derivative graph
    def init_graph_first_derivative(self):
        x = np.arange(-20, 20, 0.01)
        y = self.fn2(x)
        plt.figure(figsize=(10, 6))
        plt.plot(x, y)
        plt.xlim(-20, 20)
        plt.ylim(-20, 20)
        plt.title("Plot of the first derivative")
        plt.ylabel("Function Notation Derivative 1")
        plt.xlabel("Derivative with respect to x")
        plt.show()

    # Second derivative graph
    def init_graph_second_derivative(self):
        x = np.arange(-20, 20, 0.01)
        y = self.f_d_3(x)
        plt.figure(figsize=(10, 6))
        plt.plot(x, y)
        plt.xlim(-20, 20)
        plt.ylim(-20, 20)
        plt.title("Plot of Notation Derivative 2")
        plt.ylabel("Function Notation Derivative 2")
        plt.xlabel("Derivative with respect to x")
        plt.show()


    # Calculates the domain of the function
    def calc_domain(self):
        func_domain = (self.x + 2)**2
```

```
        func_eq = solve(func_domain, self.x)
        print("(x + 2)**2 = 0:", func_eq[0])

    # Calculates and prints the critical points of the function
    def calc_critical_points(self):
        fn = -1/((self.x + 2)**2) + 4
        self.fn_der_1 = fn.diff(self.x)
        print("d_1(f(x)):", self.fn_der_1)

    # Calculates the first derivative
    def calc_first_derivative(self):
        self.first_der_eq = solve(self.fn_der_1, self.x)
        print("d_1(f(x))=0:", self.first_der_eq)

    # This function calculates and prints the intervals where the function is + and -
    def calc_intervals(self):
        int_x = np.arange(-9999, 9999, 1)
        i = 0
        l_y = []
        for a in int_x:
            a = int(a)  # Converts to integers, if the value is of type float
            if a == -2:
                y = "NA"
                d = "NA"
            else:
                y = self.fn1(a)
                d = self.fn2(a)
            l_y.append([i, a, y, d])
            i += 1

        for i in range(0, len(l_y)):
            if l_y[i][1] == -2:
                l = ""
            elif l_y[i][3] > 0:
                l = "+"
            elif l_y[i][3] < 0:
                l = "-"
            else:
                l = ""
            l_y[i].append(l)

        y_range = []
        y_range.append(l_y[1])
        for i in range(0, len(l_y) - 1):
            if l_y[i][1] == -2:
                for j in range(-10, 10):
                    y_range.append(l_y[i + j])
        y_range.append(l_y[-1])

        result_cols = ["", "x", "y", "Derivative 1 Function Notation", "+/-"]

        function_result = pd.DataFrame(y_range, columns=result_cols)
        print(function_result)


    # Calculates the inflection points
    def calc_inflection_points(self):
        self.fn_der_2 = self.fn_der_1.diff(self.x)
        print("d_2(f(x)):", self.fn_der_2)

    # Calculates the second derivative
    def calc_second_derivative(self):
        fn_der_2_eq = solve(self.fn_der_2, self.x)
```

```
        print("d_2(f(x))=0:", fn_der_2_eq)


    # Calculates the x and y intercepts
    def calc_intercepts(self):
        fn = self.fn1(self.x)
        y_intercept = fn.subs(self.x, 0)
        x_intercept = solve(fn, self.x)
        print("f(0) (y-intercept):", y_intercept)
        print("f(x)=0 (x-intercept):", x_intercept)


    # Calculates the asymptotes
    def calc_asymptotes(self):
        fn = self.fn1(self.x)
        lim1 = limit(fn, self.x, -2)
        print("limit(f(x), x, -2):", lim1)
        fn2 = self.fn1(self.x)/self.x
        lim2 = limit(fn2, self.x, oo)
        print("limit(f(x)/x, x, oo):", lim2)
        fn3 = self.fn1(self.x)
        lim3 = limit(fn3, self.x, oo)
        print("limit (f(x)-mx, x, oo):", lim3)


def main():
    # Calls the methods defined in the class FunctionGrapher
    app = FunctionGrapher()
    app.init_graph_fn1()
    app.calc_domain()
    app.calc_critical_points()
    app.init_graph_first_derivative()
    app.calc_first_derivative()
    app.calc_intervals()
    app.calc_inflection_points()
    app.init_graph_second_derivative()
    app.calc_second_derivative()
    app.calc_intercepts()
    app.calc_asymptotes()


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


Plot of the first derivative
Function Notation Derivative 1
Derivative with respect to x


Function Notation
f(x)
x


Plot of Notation Derivative 2
Function Notation Derivative 2
Derivative with respect to x

(i) Domain of $f(x)$ is $(-\infty,-2)\cup(-2,\infty)$

(ii) Range of $f(x)$ is $(-\infty,4)$

(iii)Critical points are points where the function is defined and its derivative is zero or undefined.

$$f'(x)=\frac{2}{(x+2)^3}$$

There is **no** point where the function is defined and its derivative is zero or undefined.

(iv) Interval of decrease is

$$f'(x)=\frac{2}{(x+2)^3}<0$$

$$x\in(-\infty,-2)$$

(v) Interval of increase is

$$f'(x)=\frac{2}{(x+2)^3}>0$$

$$x\in(-2,\infty)$$

(vi)

Y-intercept is

$$y=-\frac{1}{(x+2)^2}+4 \text{ ; For y intercepts at } x=0$$

$$(0,3.75)$$

X-intercepts are

$$y=-\frac{1}{(x+2)^2}+4 \text{ ; For x intercepts at } y=0$$

$(-3/2,0)$ and $(-5/2,0)$

(vii)

An inflection point is point on the graph at which the second derivative changes sign.

$$f''(x)=\frac{d^2}{dx^2}\left(-\frac{1}{(x+2)^2}+4\right)=-\frac{1}{(x+2)^4}=0$$

As there is no solution of the above equation. Thus, there is no inflection point.

(viii)

If $f''(x)>0$ then f(x) concave upwards.

$$f''(x)=\frac{d^2}{dx^2}\left(-\frac{1}{(x+2)^2}+4\right)=-\frac{1}{(x+2)^4}>0$$

As there is no solution, there is no concave upwards.

If $f''(x)<0$ then f(x) concave downwards.

$$f''(x)=\frac{d^2}{dx^2}\left(-\frac{1}{(x+2)^2}+4\right)=-\frac{1}{(x+2)^4}$$

$$x\in(-\infty,-2)\cup(-2,\infty)$$

(ix)

Vertical asymptote is x=-2

Horizontal asymptote is y=4

```
(x + 2)**2 = 0: -2
d_1(f(x)): 2/(x + 2)**3
d_1(f(x))=0: []
              x        y Derivative 1 Function Notation +/-
0       1 -9998        4                    -2.0024e-12   -
1    9987   -12     3.99                         -0.002   -
2    9988   -11  3.98765                     -0.00274348   -
3    9989   -10  3.98438                     -0.00390625   -
4    9990    -9  3.97959                      -0.0058309   -
5    9991    -8  3.97222                     -0.00925926   -
6    9992    -7     3.96                         -0.016   -
7    9993    -6   3.9375                       -0.03125   -
8    9994    -5  3.88889                      -0.0740741   -
9    9995    -4     3.75                          -0.25   -
10   9996    -3        3                             -2   -
11   9997    -2       NA                             NA
12   9998    -1        3                              2   +
13   9999     0     3.75                           0.25   +
14  10000     1  3.88889                       0.0740741   +
15  10001     2   3.9375                         0.03125   +
16  10002     3     3.96                           0.016   +
17  10003     4  3.97222                      0.00925926   +
18  10004     5  3.97959                       0.0058309   +
19  10005     6  3.98438                      0.00390625   +
20  10006     7  3.98765                      0.00274348   +
21  19997  9998        4                           2e-12   +
d_2(f(x)): -6/(x + 2)**4
d_2(f(x))=0: []
f(0) (y-intercept): 3.75000000000000
f(x)=0 (x-intercept): [-2.50000000000000, -1.50000000000000]
limit(f(x), x, -2): -oo
limit(f(x)/x, x, oo): 0
limit (f(x)-mx, x, oo): 4

Process finished with exit code 0
```

7. The rate of growth of the profit (in millions) from an invention is approximated by the function $P'(x) = xe^{-x^2}$ where x represents time measured in years. The total profit in year two that the invention is in operation is \$25,000. Find the total profit function. Round to three decimals.

Given the profit function is $P'(x) = xe^{-x^2}$ where x is time in years. Also given that the profit in 2nd year is \$25000. That is to say;

$$P(2) = 25000$$

$$P(x) = \int xe^{-x^2}dx = -\frac{e^{-x^2}}{2} + C$$

Since $P(2) = 25000$;

$$P(2) = 25000 = -\frac{e^{-4}}{2} + C \text{ ; Thus, } C = 25000 + \frac{e^{-4}}{2}$$

Thus, $$P(x) = -\frac{e^{-x^2}}{2} + \frac{e^{-4}}{2} + 25000 = \frac{e^{-4} - e^{-x^2}}{2} + 25000 = -\frac{e^{-x^2}}{2} + 25000.009$$

8. For a certain drug, the rate of reaction in appropriate units is given by

$$R'(t) = \frac{6}{t+1} + \frac{1}{\sqrt{t+1}}$$

where t is time (in hours) after the drug is administered. Find the total reaction to the drug from 1 to 8 hours after it is administered. Round to two decimal places.

Reaction during an interval is:

$$R(t) = \int_{T_1}^{T_2} R'(t)dt$$

$$R(t) = \int_1^8 \left(\frac{6}{t+1} + \frac{1}{\sqrt{t+1}}\right)dt = \left[6\ln(t+1) + 2\sqrt{t+1}\right]_1^8 = 6\ln\left(\frac{9}{2}\right) + 6 - (2)^{\frac{3}{2}} \approx 12.196$$

The total reaction to the drug from 1 to 8 hours after it is administered is = **12.20**.

9. Determine if the following function is a probability density function on $[0, \infty)$.

$$f(x) = \begin{cases} {}^{x^3}/_{8}, if\ 0 \le x \le 2 \\ {}^{4}/_{x^3}, if\ x > 2 \end{cases}$$

If it is a probability density function, then calculate $P(1 \le x \le 5)$ and round to four decimal places. If it is not, explain why.

The probability density function (PDF) is the PD of a continuous random variable. The cumulative density function (CDF) is the cumulation of the probability of all the outcomes upto a given value.

In other word, the CDF is the probability that the RV can take any value less than or equal to X. If we assume that the RV X can take values from $-\infty$ to ∞, then theoretically,

$$F(X) = \int_{-\infty}^{X} f(x)dx$$

$$P(1 \le x \le 5) = \int_0^2 f(x)dx + \int_2^5 f(x)dx$$

$$= \int_0^2 \frac{x^3}{8}dx + \int_2^5 \frac{4}{x^3}dx$$

$$= \left[\frac{x^{3+1}}{(3+1)8}\right]_0^2 + 4\left[\frac{x^{(-3+1)}}{(-3+1)}\right]_2^5$$

$$= \left[\frac{x^4}{32}\right]_0^2 + \left[-\frac{2}{x^2}\right]_2^5$$

$$= \frac{2^4}{32} - \frac{2}{5^2} + \frac{2}{2^2}$$

$$= \frac{2^4 + 16}{32} - \frac{2}{50}$$

$$= 1 - \frac{2}{25} = \frac{23}{25}$$

10. Researchers have shown that the number of successive dry days that occur after a rainstorm for a particular region is a random variable that is distributed exponentially with a mean of 9 days. **Using Python**, determine the (separate) probabilities that 13 or more successive dry days occur after a rainstorm, and fewer than 2 dry days occur after a rainstorm. Round the probabilities to four decimal places.

```
from sympy import *


class Probability():

    def __init__(self):
        self.x = Symbol("x") # Symbol x has been defined

    # Initializes the random variable  which is distributed exponentially with a mean
of 9 days
    def random_variable(self):
        self.rv = float(1/9)*exp(-self.x/9)

    # Calculates the probabilities of dry days after a rainstorm. Takes days as
parameters
    def calculate(self, param1, param2):
        prob = integrate(self.rv,(self.x, param1, param2))
        prob_eval = prob.evalf()
        return prob_eval



def main():
    app = Probability()     # Creating the class object
    app.random_variable()   # Initializing the random variable

    # Calculates to find the probability that 13 or more successive dry days occur
after a rainstorm
    val_1 = app.calculate(13, oo)
    prob_1 = format(val_1, ".4f")
    print("P(X>=13): ", prob_1)

    # Calculates to find the probability that less than 2 dry days occur after a
rainstorm
    val_2 = app.calculate(0, 2)
    prob_2 = format(val_2, ".4f")
    print("P(X<2): ", prob_2)



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

The probability density function (pdf) of an exponential distribution is:

$$f(x) = \lambda e^{-\lambda x} \quad x \geq 0$$
$$= 0 \qquad \mathbf{x < 0}$$

The mean or expected value of an exponentially distributed random variable X with rate parameter λ is given by:

$$E[X] = \frac{1}{\lambda}$$

Given that $E[X] = \frac{1}{\lambda} = 9$; thus $\lambda = \frac{1}{9}$

The cumulative distribution function is given by;

$$P(x \leq 13) = \int_{-\infty}^{13} f(x)dx = \int_{-\infty}^{13} \frac{1}{9} e^{-\frac{x}{9}} dx = 1 - e^{-\frac{13}{9}} \quad \textbf{for x} \geq \textbf{13}$$

$$P(x > 13) = 1 - (1 - 0.2359) = 0.2359$$

Probability that 13 or more successive dry days occur after a rainstorm = 0.2359

Probability that fewer than 2 dry days occur after a rainstorm

=Probability that 1 dry day occurs after a rainstorm

$$= P(x < 2)$$
$$= (1 - e^{-2/9})$$
$$= 1 - e^{2/9}$$
$$= 1 - 0.8007 = 0.1993$$