**Project Title**

By

# Kshama Purohit (L049)

Under the supervision of

## Name of Mentors

(Designation, CSDS 311 Program, MPSTME)

Name of the Chairperson

( )

**SVKM’s NMIMS University**

(Deemed-to-be University)

****

**MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING**

**Vile Parle (W), Mumbai-56**

**2022-23**

**Table of Contents**

1. **Introduction**
   1. Background of the Project
   2. Motivation and Scope of the Report
   3. Problem Statement
2. **Literature Review**
   1. Introduction to the Overall Topic
   2. Exhaustive Literature Review
3. **Methodology and Implementation**
   1. Block Diagram, Flochart, Algorithm
   2. Code using Python 3.10.5
   3. Output
4. **Result and Anaysis**
5. **References**

**Chapter 1**

**Introduction**

* 1. **Background of the project topic**

This combined mini project is an accumulation of a business statistics concept, analyzed using python and visualized through SAS.

The idea behind this assignment was to create a functioning model for optimizing the cost incurred by various campaigns launched by a company with regards to advertisement, using Facebook’s geographically targeted sample data.

* 1. **Motivation and scope of the report**

A business’ true goal is to earn profit and maximize sales. That is the very outcome that pushes entrepreneurs to take risks and businessmen to play with uncertainty.

A way of generating maximum profit is to cut down costs. To understand the ins and outs of cash flows of a business, to analyze expenditures and revenues This can be done by channeling money into better, more profitable avenues.

This is the motivation behind our project – to ensure that cash outflows are in the right direction and impact the business in a positive, profitable manner.

* 1. **Problem statement**

The BOD meeting of Facebook revealed that the directors are not happy with the number of returns they're receiving based off all the advertisement campaigns.

Our job as data scientists is to collect, analyze, optimize and then visualize the data so as to make sure that the company is able to minimize its costs from the advertisements.

**Chapter 2**

**Literature survey**

* 1. **Introduction to overall topic**

This project aids the company Facebook in allocating funds to the right advertisement campaigns, to allow them to minimize costs and maximize profits.

We have utilized the concept of ***Linear Programming Problem*** to optimize the costs incurred by Facebook in hopes to minimize them.

* 1. **Exhaustive literature survey**

Linear programming is a part of applied mathematics that deals with solving optimization problems. Linear programming problems consist of a linear cost function (consisting of a certain number of variables) which is to be minimized or maximized subject to a certain number of constraints. The constraints are linear inequalities of the variables used in the cost function (Schulze, 2000).

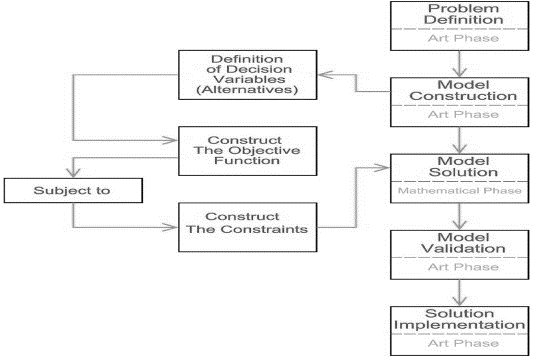


Figure 1: The basic algorithm of a linear programming problem

The basic terms used in a linear programming model are as follows:

***Decision Variables***: The variables that will affect the output are the decision variables. Before we can solve any problem, we have to figure out what the decision variables are.

***Objective******Equation***: It is defined as the reason for making a choice. This equation can either be minimized or maximized.

***Constrain*ts**: The constraints are the things that limit or stop the decision variables from doing what they want to do. Most of the time, they limit how much the decision variables matter.

The constraints specified by the problem set a boundary for the solution. It allows us to find an optimized solution within the range of the given constraints.

Regions outlined by the constraints can be of 3 types:

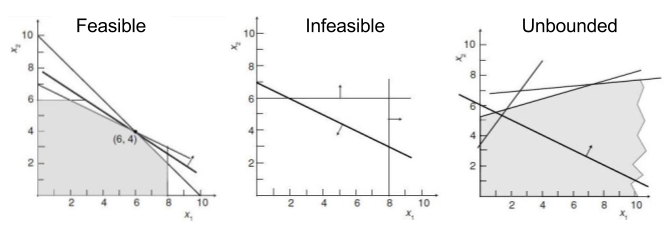
* *Feasible Region*: The given programming model has an optimal solution that exists within the range of the constraints.
* *Infeasible Region*: The given programming model has no solution thatexists within the specified range of the constraints. There exists no solution for which all the constraints are satisfied.
* *Unbounded Region*: The given programming model is unbounded if there aren't enough constraints on the cost function to restrain it enough for every possible solution to be followed by another feasible solution that yields a better result.

Figure 2: The three types of linear programming models

***Non-negativity constraints***: For all linear programs, the decision variables should never have negative values. This means that the values of the decision variables should be greater than or equal to 0 (Avcontentteam, 2020).

The basic geometric representation of a linear programming model is as follows:

z = a0x0+a1x1+a2x2…anxn

subject to,

c1 </>= x0b0+x1b1+x2b2 …xnbn

c2 </>= x0d0+x1d1+x2d2 …xndn

**.**

**. .**

cn </>= x0z0+x1z1+x2z2 …xnzn

Linear Programming is widely used in production planning and scheduling

Problems. The biggest advantage of linear programming as an optimization method is that it always achieves the optimal solution if one exists. Using this concept, we have solved the given problem statement, i.e. to optimize (in our case, minimize) the costs incurred by Facebook.

**Chapter 3**

**Methodology and Implementation**

**3.1 Block Diagram, Flowchart and Algorithm**

**3.2 Code using Python (3.10.5)**

import matplotlib.pyplot as plt #plotting various graphs

import pandas as pd #data processing, CSV file I/O (e.g. pd.read\_csv)

from pulp import LpProblem,LpMinimize,LpVariable,LpStatus,LpInteger #linear programming library

x=lambda x:print('\n')

df=pd.read\_csv("C:\sample data.csv")

print(df)

x(1)

print(df.describe())  #describing the dataset

x(1)

print("[ID 916, ID 936, ID 1178]")

x(1)

print('([impressions, clicks, total conversions]\n')

x(1)

class lpp(): #initializing class

    var=[[],[],[]]

    def valfind(self,id,i,obj):

        #function to find and store the computational values for the linear programming model

        obj.var[i].append((df.loc[df['company campaign ID'] == id,'impressions'].mean())) #computing the average of the impression for each campaign ID

        obj.var[i].append((df.loc[df['company campaign ID'] == id,'clicks'].mean())) #computing the average of the clicks for each campaign ID

        obj.var[i].append((df.loc[df['company campaign ID'] == id,'total conversions'].mean())) #computing the average of the total conversions for each campaign ID

lpp.valfind=classmethod(lpp.valfind)

lppvar=lpp() #creating 'lppvar' as an instance/object of class 'lpp'

lppvar.valfind(916,0,lppvar) #invoking function 'valfind()' of class 'lpp' using object 'lppvar' to store average values for ID 916

lppvar.valfind(936,1,lppvar) #invoking function 'valfind()' of class 'lpp' using object 'lppvar' to store average values for ID 936

lppvar.valfind(1178,2,lppvar) #invoking function 'valfind()' of class 'lpp' using object 'lppvar' to store average values for ID 1178

print(lppvar.var) #printing values of the variable 'var' that stores all the values for the programming problem

x(1)

a=df.loc[df['company campaign ID']==916] #splitting the dataset for ID 916

print(a)

x(1)

b=df.loc[df['company campaign ID']==936] #splitting the dataset for ID 936

print(b)

x(1)

c=df.loc[df['company campaign ID']==1178] #splitting the dataset for ID 1178

print(c)

x(1)

y=lambda y:(df.loc[df['company campaign ID'] == y,'spent'].mean()) #lamda function to find the average cost for each campaign ID

a\_cost=int(y(916))

b\_cost=int(y(936)) #calling the lamda function for ID 936

c\_cost=int(y(1178))

const=[]

def intake():

    #function to accept and store the constraint values

    temp=0

    check=1 #flag variable to check if entered value meets the if condition or not

    temp=int(input("enter the least value of impressions that you wish to obtain: "))

    while(check!=0):

        if temp>=df['impressions'].mean(): #checking if the entered constraint value is atleast equal to or more than the average value of the entire dataset

            const.append(temp) #if condition is true, append list 'const' with entered constraint value

            check=0 #change value of flag to instruct the code to exit the while loop

        else:

            temp=int(input(('value isnt in the range, enter again: '))) #re-enter constraint value

            check=1 #flag value remains the same, so the while loop keeps running and the user keeps entering until the condition is met

    temp=int(input("enter the least value of clicks that you wish to obtain from the viewer: "))

    while(check!=1):

        if temp>=df['clicks'].mean():

            const.append(temp)

            check=1

        else:

            temp=int(input(('value isnt in the range, enter again: ')))

            check=0

    temp=int(input("enter the least value of conversions that you wish to obtain: "))

    while(check!=0):

        if temp>=df['total conversions'].mean():

            const.append(temp)

            check=0

        else:

            temp=int(input(('value isnt in the range, enter again: ')))

            check=1

intake()

def linearproblem():

    #function to solve the linear programming model

    problem = LpProblem("AD Conversions", LpMinimize) #creating the 'problem' variable to store the optimization equation and the constraints

    a1 = LpVariable("ID\_916", lowBound=0, cat=LpInteger) #creating lpp variables

    b1 = LpVariable("ID\_936", lowBound=0, cat=LpInteger)

    c1 = LpVariable("ID\_1178", lowBound=0, cat=LpInteger)

    problem += a1\*a\_cost + b1\*b\_cost + c1\*c\_cost, 'ObjectiveFunction'

    for i in range (0,3):

        problem += (lppvar.var[0][i])\*a1 + (lppvar.var[1][i])\*b1 + (lppvar.var[2][i])\*c1>=(const[i]) #creating the cnstraint equations

    x(1)

    print(problem)

    x(1)

    #print(LpStatus[problem.status])

    x(1)

    problem.solve()

    x(1)

    #print(LpStatus)

    #print(LpStatus[problem.status])

    x(1)

    print("The amount of money to be spent on ID 916 is: ", a1.varValue)

    x(1)

    print("The amount of money to be spent on ID 936 is: ", b1.varValue)

    x(1)

    print("The amount of money to be spent on ID 1178 is: ", c1.varValue)

linearproblem()

plt.subplot(1,2,1)

plt.scatter(df["impressions"], df["total conversions"])

plt.xlabel("impressions")

plt.ylabel("total conversions")

plt.subplot(1,2,2)

plt.scatter(df["impressions"], df["approved conversions"])

plt.xlabel("impressions")

plt.ylabel("approved conversions")

plt.show()

plt.bar(df['gender'],df['roas'])

plt.ylabel("gender")

plt.xlabel("ROAS")

plt.show()

plt.plot(df['roas'],df['spent'])

plt.ylabel("amount spent")

plt.xlabel("ROAS")

plt.show()

plt.subplot(1,3,1)

plt.boxplot(df['spent'])

plt.xlabel("amount spent")

plt.subplot(1,3,2)

plt.boxplot(df['roas'])

plt.xlabel("ROAS")

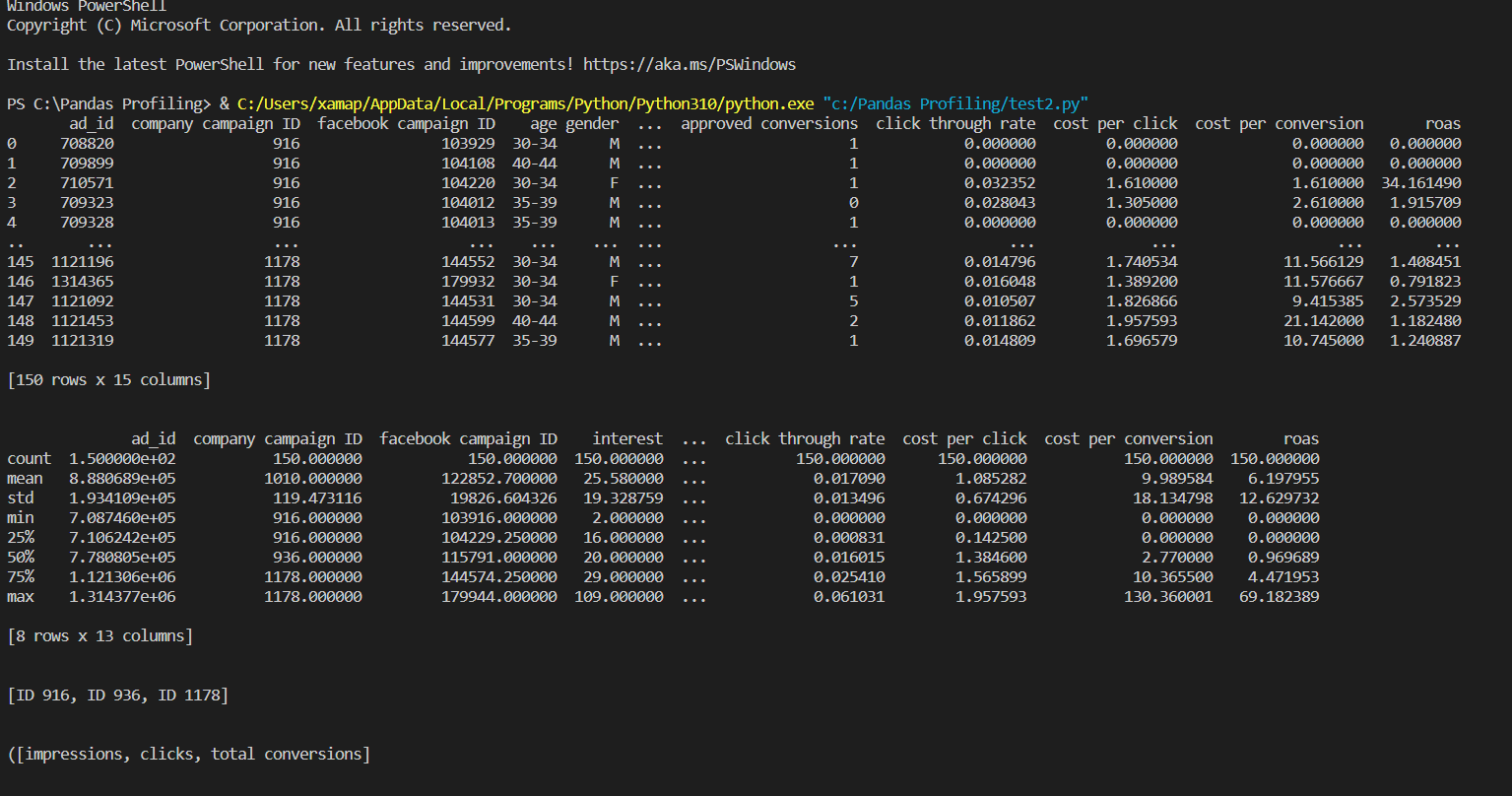
plt.subplot(1,3,3)

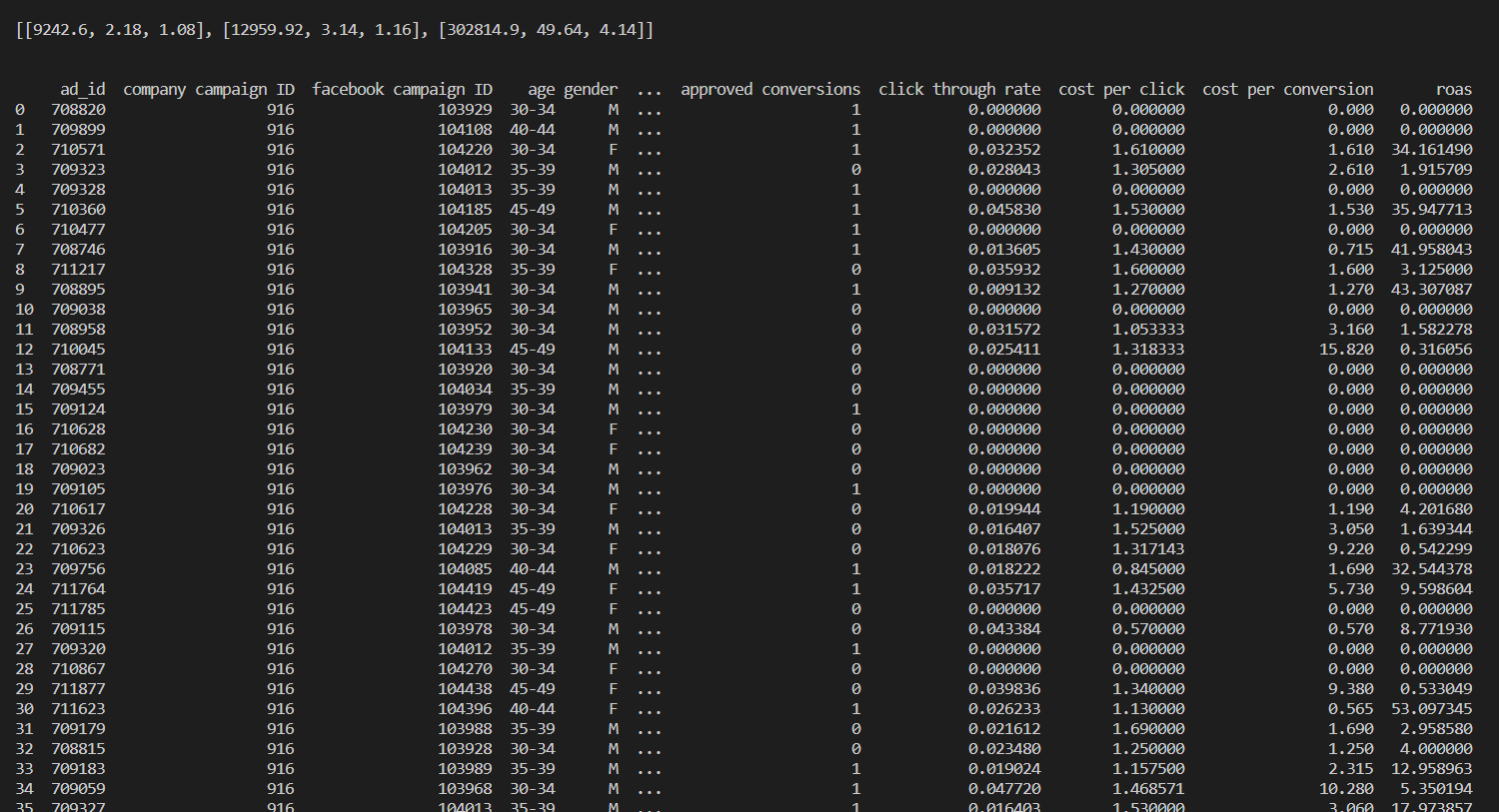
plt.boxplot(df['total conversions'])

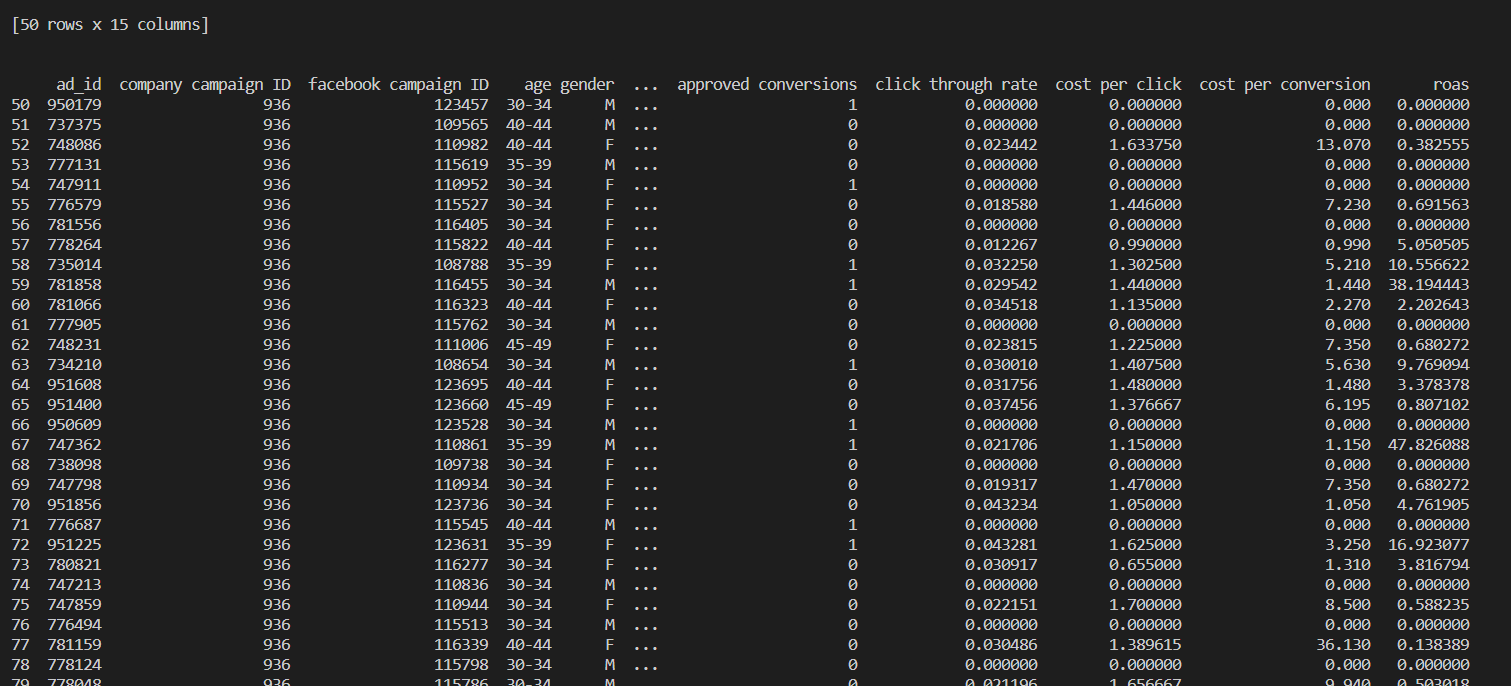
plt.xlabel("total conversions")

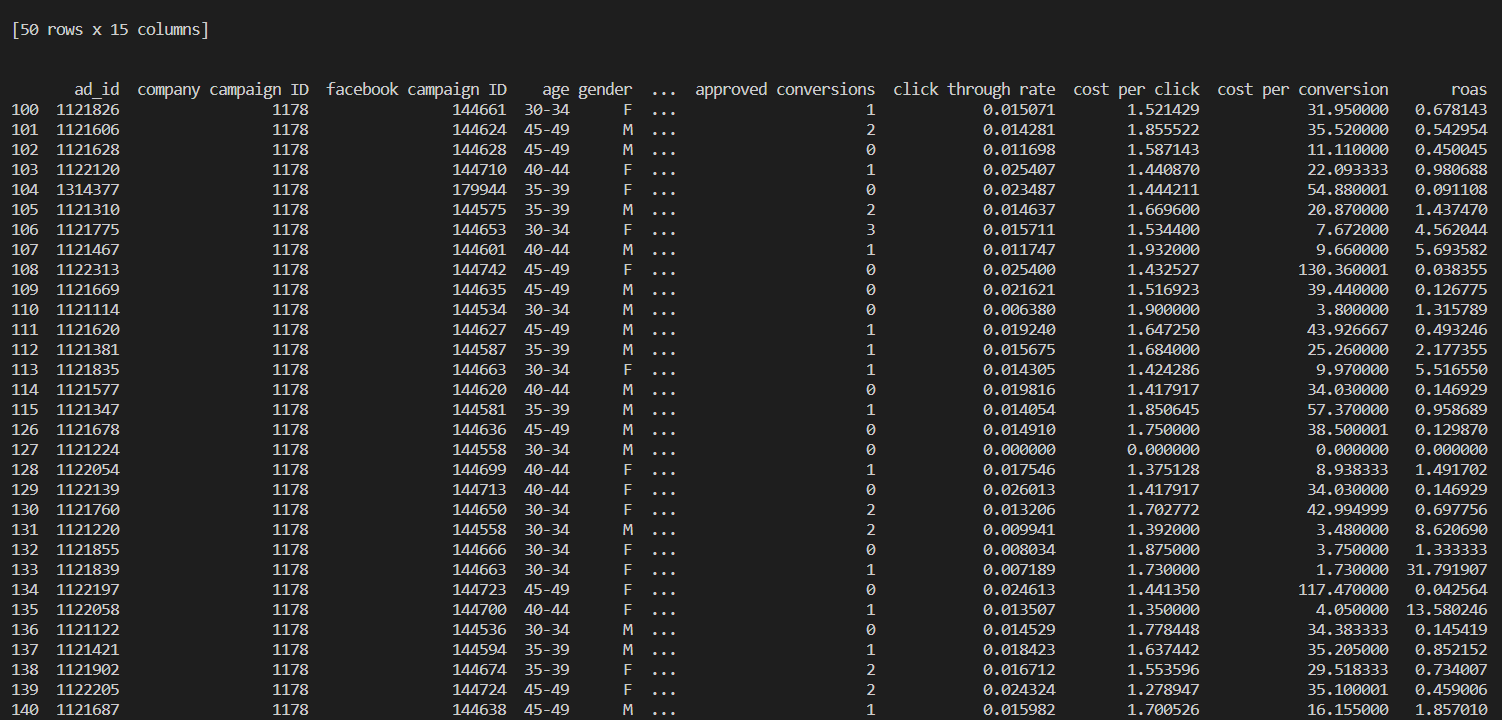
plt.show()

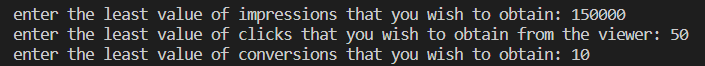
**3.3 Output**

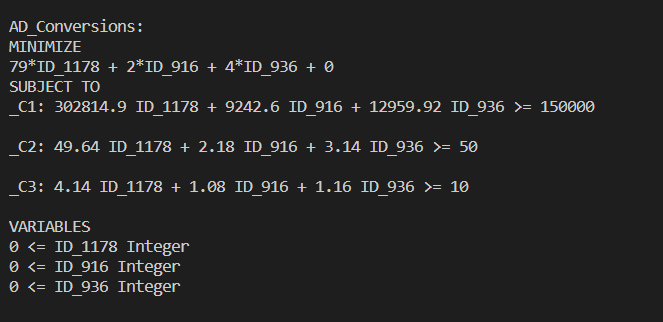
****

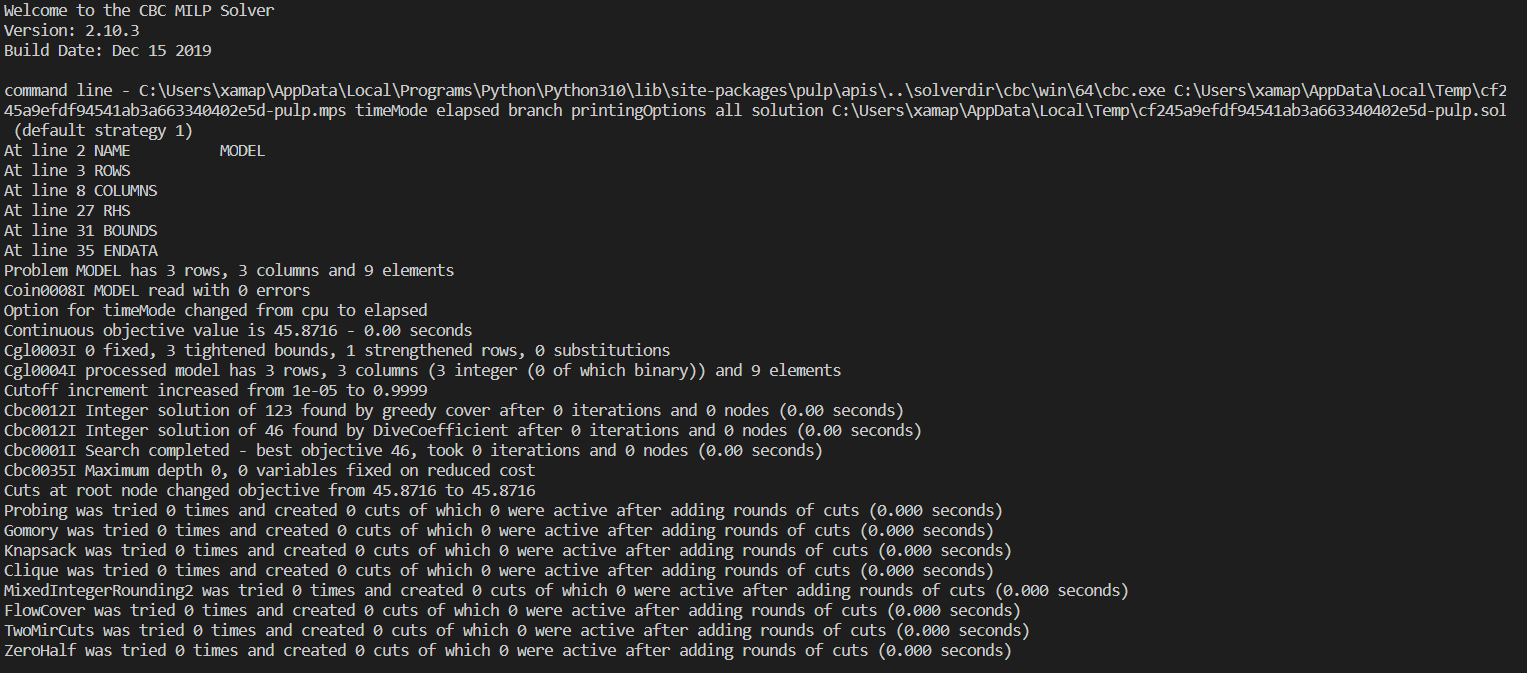
****

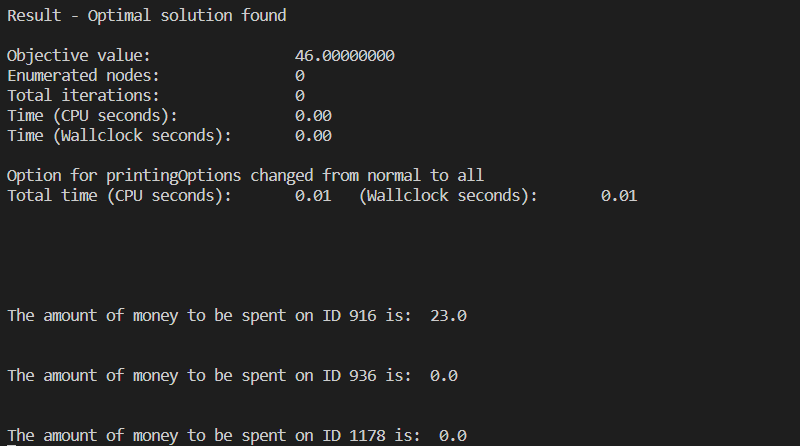
****

****

****

****

****

****

**Chapter 4**

**Results and Analysis**

This shall include an evaluation and investigation carried out.

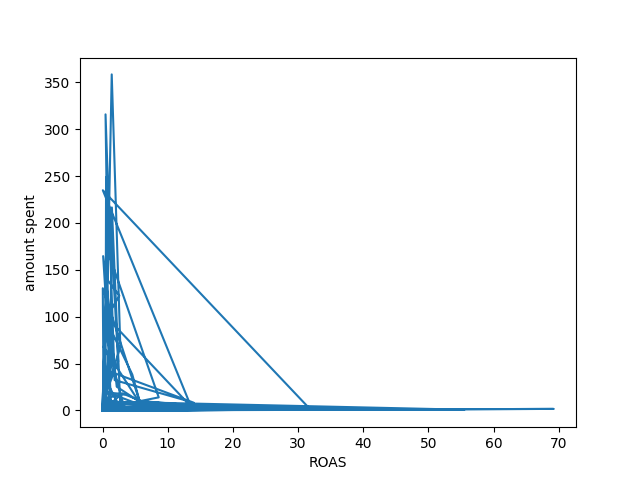
****

Figure 3: Amount Spent vs ROAS

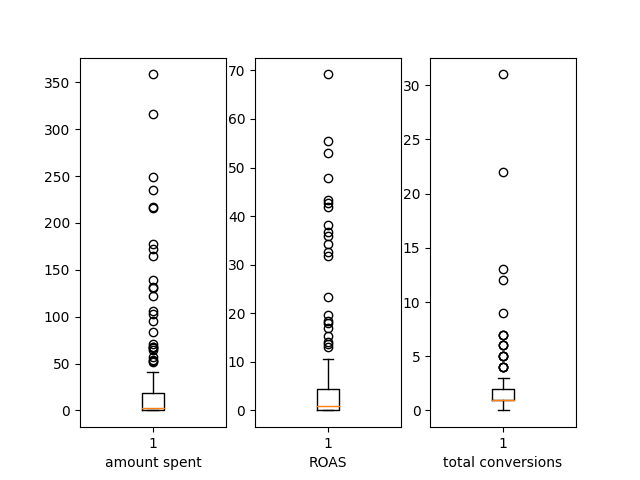
****

Figure 4: Amount Spent vs ROAS vs Total Conversions

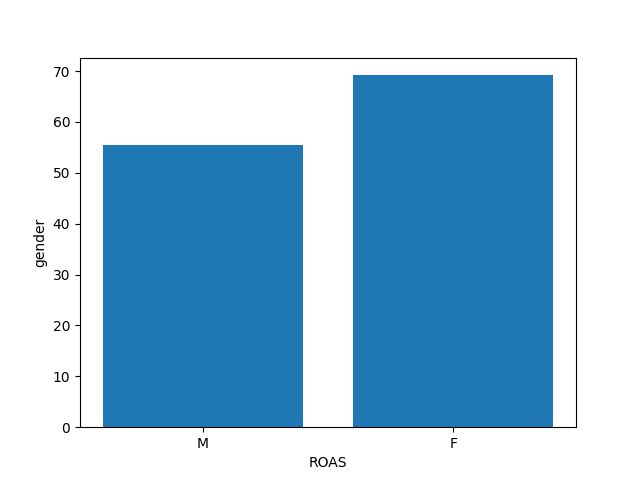
****

Figure 5: ROAS by Gender

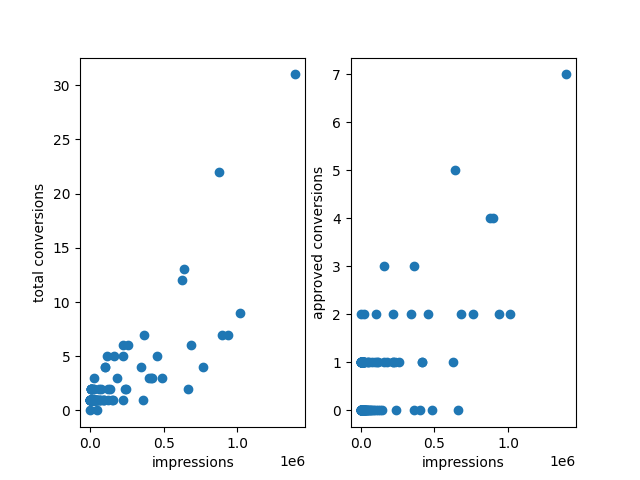
****

Figure 6: Total Conversions vs Approved Conversions by Impressions

**References**

1. Avcontentteam. (2020, April 1). *Linear Programming: Applications of Linear Programming*. Analytics Vidhya. Retrieved October 16, 2022, from https://www.analyticsvidhya.com/blog/2017/02/lintroductory-guide-on-linear-programming-explained-in-simple-english/
2. Banerjee, R. (2021, February 3). *Basic linear programming in Python with Pulp*. Medium. Retrieved October 12, 2022, from https://towardsdatascience.com/basic-linear-programming-in-python-with-pulp-d398d144802b
3. by admin, W. (n.d.). *Limiting factor analysis*. MASOMO MSINGI PUBLISHERS. Retrieved October 9, 2022, from https://masomomsingi.com/limiting-factor-analysis/
4. Du, D.-Z., Pardalos, P., Hu, X., & Wu, W. (2022). Linear Programming. *Introduction to Combinatorial Optimization*, 129–174. https://doi.org/10.1007/978-3-031-10596-8\_6
5. Gokagglers. (2017, September 26). *Sales conversion optimization*. Kaggle. Retrieved October 12, 2022, from https://www.kaggle.com/datasets/loveall/clicks-conversion-tracking
6. Kevinbinz. (2017, April 3). *Feasible solution*. Fewer Lacunae. Retrieved October 12, 2022, from https://kevinbinz.com/tag/feasible-solution/
7. Navlani, A., & Pandey, P. (2022, April 30). *Solving linear programming using Python Pulp*. Machine Learning Geek. Retrieved October 12, 2022, from https://machinelearninggeek.com/solving-linear-programming-using-python-pulp/
8. *Optimization with pulpℑ*. Optimization with PuLP - PuLP 2.6.0 documentation. (n.d.). Retrieved October 12, 2022, from https://coin-or.github.io/pulp/
9. *Pulp*. PyPI. (n.d.). Retrieved October 12, 2022, from https://pypi.org/project/PuLP/
10. *Sign in*. RPubs. (n.d.). Retrieved October 12, 2022, from https://rpubs.com/nicokaswalder/614266