Optimization Project 1
Project 1: Linear Programming

Group 20: Amanda Nguyen (atn844), Ryan Lee (rl32227), Aashi Aashi (aa92533), Samarth Mishra (sm79247)

OBJECTIVE

In a recent study, marketing budgets were discovered to represent 11 percent of a company's total budget. However, marketing effectiveness seems to vary significantly between companies. There can be companies that cut marketing spending as a result of ineffective advertising, while other companies may increase their spending as a result of positive feedback. A potential reason for large variation in marketing success derives from marketing budget allocations.

We're trying to use linear programming in order to generate a simple marketing budget allocation strategy. We must find an optimal budget spread among multiple marketing mediums based on ROI percentages derived from consultants to fit the $10 million budget.

As data scientists in the marketing department, we must decide on an optimal marketing budget allocation strategy. The objective is to maximize expected ROI in millions of dollars. Return data for each marketing medium will be extracted from two separate consulting firm estimates.

SPECIFICS

The provided information and constraints are as follows:

1. Marketing budget of $10 million
2. The amount invested in print and TV advertising should be no more than the amount spent on Facebook and Email.
3. The total amount used in social media should be at least twice of SEO and AdWords.
4. For each platform, the amount invested should be no more than $3M.

CASE 1: USING FIRST CONSULTING ROI VALUES

Platform	Print	TV	SEO	AdWords	Facebook	LinkedIn	Instagram	Snapchat	Twitter	Email
ROI	3.1%	4.9%	2.4%	3.9%	1.6%	2.4%	4.6%	2.6%	3.3%	4.4%

For this first case, we began by reading in the ROI_data csv file in order to retrieve the data for marketing medium return percentages. We then acknowledged the constraints pertinent to finding the optimal marketing budget allocation strategy. From this, we were able to begin forming our linear program.

The first step performed included setting our objective, which was to maximize the ROI by choosing the proper allocation of the budget into each of the ten marketing mediums acquired by reading in the csv file.

Then, we used the restrictions in place to create a constraint matrix that our objective was subject to. The first constraint (budget) required the sum of the marketing mediums to equal the allocated $10 million. The second constraint subjected the amount spent on Facebook and Email to be greater than or equal to that of print and TV advertising. The third constraint mandated the amount used on social media (LinkedIn, Facebook, Instagram, Snapchat, Twitter) be greater than or equal to twice the amount of SEO and AdWords. The last constraint enforced all marketing medium investments be less than or equal to $3 million.

```
# Read ROI csv
ROI = pd.read_csv('ROI_data.csv')

# Objective Vector
roi_list = list(ROI.iloc[0]) # create a list of values based off the first row in the CSV
obj= np.array(roi_list[1:11]) # create the objective based off the list of ROIs
obj[0:] += 1

# Constraint matrix
A = np.zeros((13,10))
A[0,:] = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1] # constraint 1: sum of all mediums must be less or equal to $10M
A[1,:] = [1, 1, 0, 0, -1, 0, 0, 0, 0, -1] # constraint 2: print and TV investments less than Facebook and Email
A[2,:] = [0, 0, 2, 2, -1, -1, -1, -1, -1, 0] # constraint 3: social media investments at least twice SEO and AdWords
A[3:13,: ] = np.identity(10) # constraint 4: investment amount for all platforms no more than $3M

# Limits on all constraints
b = np.array([10, 0, 0, 3, 3, 3, 3, 3, 3, 3, 3, 3])

# All constraints are less than or equal to
sense = np.array(['<'] * 13)
```

Together, we were able to use this information to form our inputs for our linear programming model, which employed Gurobi in order to find the optimal budget allocation.

```
# Initialize an empty model
ROIModel = gp.Model()

# Tell the model how many variables there are
ROImodx = ROIModel.addMVar(len(obj))

# Add the constraints to the model
ROIModelCon = ROIModel.addMConstrs(A, mktx, sense, b)

# Add the objective to the model
ROIModel.setMObjective(None, obj, 0, sense=gp.GRB.MAXIMIZE)

# Tell Gurobi to be quiet
ROIModel.Params.OutputFlag = 0

# Run the program
ROIModel.optimize()
```

Result:

The optimal ROI in dollars is: **$10,456,000.**

```
ROIModel.objVal # optimal ROI in millions of dollars
```

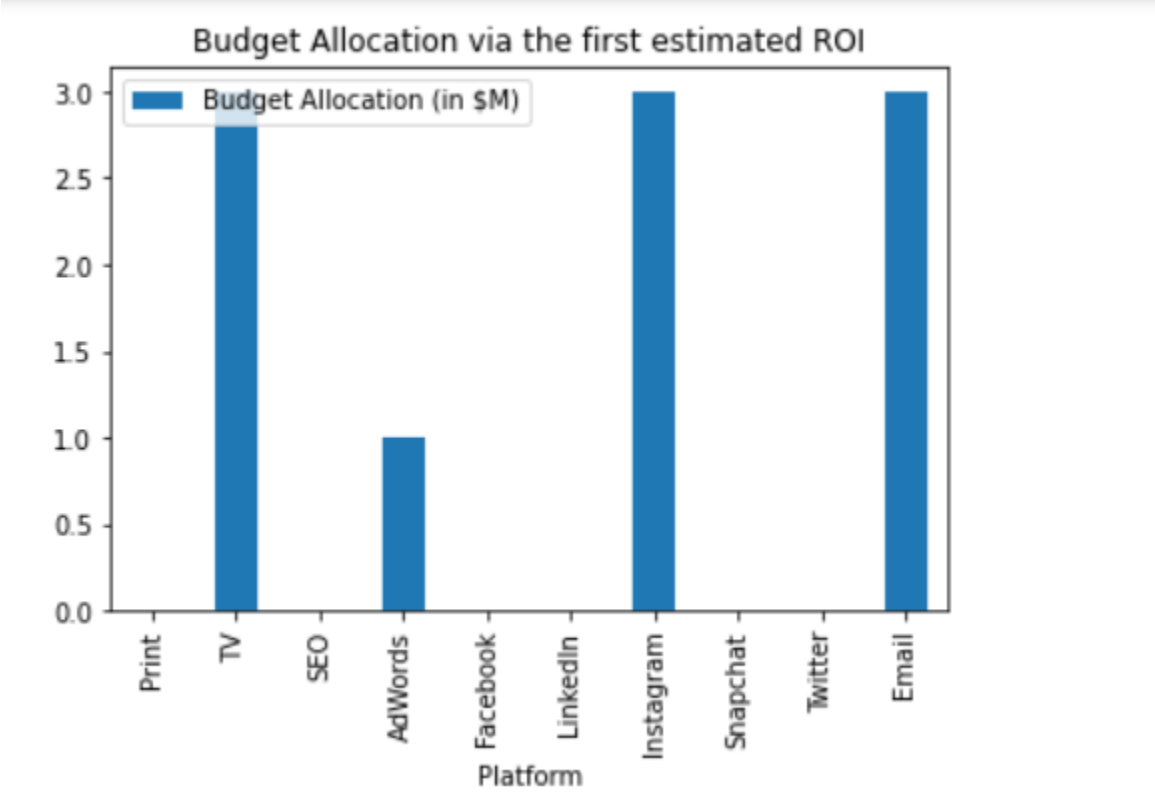
```
10.456
```

The allocation for each marketing medium was as follows:

```
ROIModX.x # how many millions of dollars to invest in platform
```

```
array([0., 3., 0., 1., 0., 0., 3., 0., 0., 3.])
```

Platform	Print	TV	SEO	AdWords	Facebook	LinkedIn	Instagram	Snapchat	Twitter	Email
$ invested	$0.0	$3M	$0.0	$1M	$0.0	$0.0	$3M	$0.0	$0.0	$3M



CASE 2: SECOND CONSULTING FIRM & COMPARISON OF OPTIMAL OPTIONS

Platform	Print	TV	SEO	AdWords	Facebook	LinkedIn	Instagram	Snapchat	Twitter	Email
ROI	4.9%	2.3%	2.4%	3.9%	4.4%	4.6%	2.6%	1.9%	3.7%	2.6%

Using the same constraints and objectives, we constructed the second model in identical fashion to the first, except using the ROI values from the second row in the ROI_data csv dataset.

```
# Creating the objective vector
NewROI = list(SecondROIdata.iloc[1])
obj2= np.array(NewROI[1:11])
obj2[0:] += 1

# Generating the constraint matrix
C = np.zeros((13,10))
C[0,:] = [1,1,1,1,1,1,1,1,1,1] # budget constraint
C[1,:] = [1,1,0,0,-1,0,0,0,0,-1] # print and TV constraint
C[2,:] = [0,0,2,2,-1,-1,-1,-1,-1,0] # social media constraint
C[3:13,:] = np.identity(10)

# Limits on the constraints
d = np.array([10,0,0,3,3,3,3,3,3,3,3,3,3])

# All constraints are less than or equal constraints
sense = np.array(['<'] * 13)
```

Once again, with these inputs, we ran the Gurobi program in order to find the new optimal allocation portfolio.

```
# Initialize an empty model
SecondROIModel = gp.Model()

# Tell the model how many variables there are
SecondROIModX = SecondROIModel.addMVar(10)

# Add the constraints to the model
Second ROIModCon = SecondROIModel.addMConstrs(C, SecondROIModX, sense, d)

# Add the objective to the model.
SecondROIModel.setMObjective(None,obj_2,0,sense=gp.GRB.MAXIMIZE)

# Tell gurobi to shut up!!
SecondROIModel.Params.OutputFlag = 0

# Run the model
SecondROIModel.optimize()
```

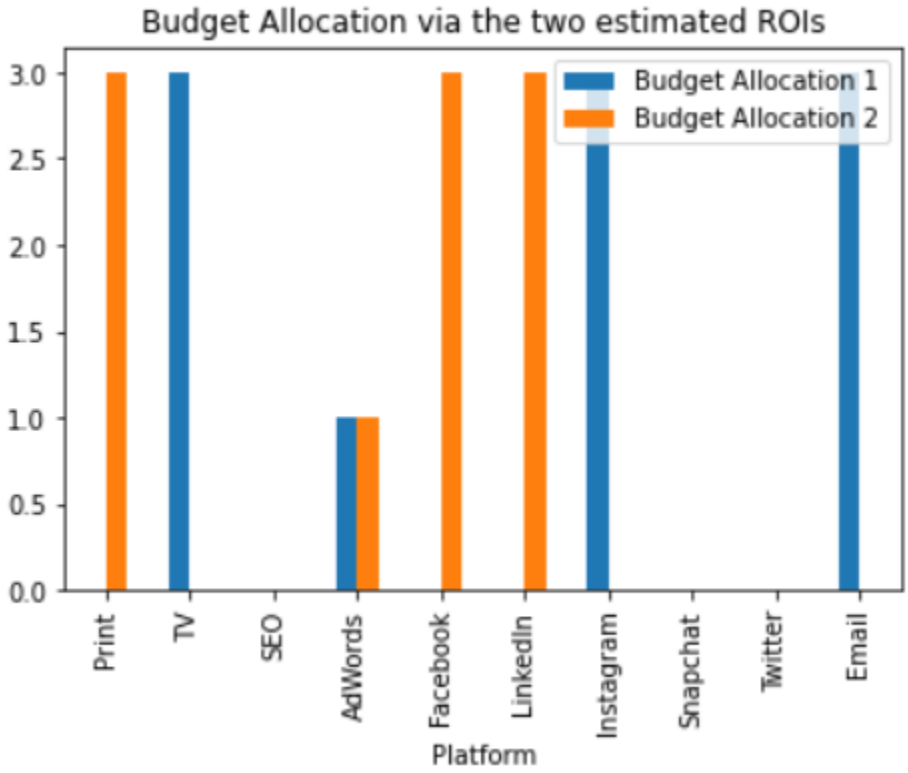
Results:
The optimal ROI in dollars is: **$10,456,000.**

The allocation for each marketing medium was as follows:

Platform	Print	TV	SEO	AdWords	Facebook	LinkedIn	Instagram	Snapchat	Twitter	Email

$ invested	$3M	$0.0	$0.0	$1M	$3M	$3M	$0.0	$0.0	$0.0	$0.0

Comparing the two allocation budgets:



It can be seen that the allocations are not the same.
Assuming the first ROI data is correct, using the second allocation the objective relative to the optimal objective (the one that uses the first ROI data and first allocation) is lower by $0.204 million.

Instead, if we were to use the first allocation when the second ROI data was correct, the objective was lower than the optimal objective by $0.193 million.

We believe the third constraint to be useful as it helps mitigate the consequences of incorrectly choosing the most appropriate ROI data. The maximum difference between the optimal solution and the feasible solutions found was $0.204 million. Although this difference is not desirable, the consequences of allocating more of the budget to a platform based on a single set of ROI values, when the actual ROI is much different (investing all $10M of budget into LinkedIn assuming ROI is 4.6% when it is actually 2.4%) would be much more severe. In essence, removing the third constraint changes the original problem statement, which would have drastic implications

on the optimal solution. Understanding this, we can make the assertion that the third constraint acts as a binding constraint and is therefore necessary to maintain.

Illustrating optimal allocation changes based on variations in ROI data:

	Lower Limit	Starting Point	Upper Limit
0	-inf	1.031	0.049
1	0.039	1.049	0.062
2	-inf	1.024	0.039
3	0.033	1.039	0.046
4	-inf	1.016	0.029
5	-inf	1.024	0.039
6	0.039	1.046	inf
7	-inf	1.026	0.039
8	-inf	1.033	0.039
9	0.029	1.044	inf

From the dataframe, one can recognize that the amount by which the marketing medium ROI fluctuates has no implication on the optimal solution, as it remains constant. An 'inf' value indicates the amount allocated to the respective medium is already its maximum value of $3 million. Therefore, increasing the ROI in that instance would have no effect as it is already maximized. However, the mediums that had $3 million invested in them do not have a lower limit for ROI. A negative 'inf' value implies that there was no capital allocated to the medium and therefore decreasing the ROI of those mediums to zero would not have influence on the allocation of the optimal budgets. The mediums do however have an upper limit for ROI.

SCENARIO 3: OPPORTUNITY TO REINVEST ½ OF RETURNS

Upon gaining permission to reinvest half of the returns while still following the original constraints, we must find the optimal allocation for each month.

First we read the monthly ROI data from file 'roi_mat.csv'. We updated the dataframe to net values post returns.

```
# Reading the expected monthly ROI data for the different platforms from 'roi_mat.csv'
roi_by_month=pd.read_csv("roi_mat.csv",index_col='Unnamed: 0')
# Updating values to actual returns
return_by_month=roi_by_month.apply(lambda x: (100+x)/100)
return_by_month
```

	Print	TV	SEO	AdWords	Facebook	LinkedIn	Instagram	Snapchat	Twitter	Email
January	1.040	1.036	1.024	1.039	1.030	1.035	1.036	1.0225	1.035	1.035
February	1.040	1.039	1.027	1.038	1.043	1.032	1.027	1.0180	1.037	1.035
March	1.035	1.029	1.031	1.038	1.024	1.041	1.037	1.0260	1.042	1.025
April	1.038	1.031	1.024	1.044	1.024	1.038	1.037	1.0250	1.036	1.029
May	1.035	1.032	1.019	1.034	1.027	1.027	1.039	1.0220	1.045	1.039
June	1.040	1.032	1.027	1.034	1.034	1.030	1.045	1.0210	1.038	1.041
July	1.039	1.036	1.020	1.044	1.039	1.037	1.043	1.0180	1.040	1.038
August	1.042	1.033	1.028	1.042	1.020	1.037	1.036	1.0150	1.044	1.043
September	1.041	1.028	1.025	1.042	1.029	1.037	1.028	1.0250	1.040	1.034
October	1.030	1.030	1.031	1.046	1.031	1.033	1.032	1.0230	1.025	1.032
November	1.048	1.033	1.027	1.041	1.029	1.036	1.042	1.0300	1.031	1.041
December	1.048	1.040	1.019	1.037	1.042	1.036	1.026	1.0290	1.036	1.037

To get the monthly optimal allocation for each of the platforms, we will have to loop in through all the months forming separate optimization models for each.
Our constraint matrix has remained the same as the previous two scenarios. However, the monthly budget keeps changing since we are reinvesting half of the returns from the previous month.

Please find below the code snippet where we loop in through all the months forming separate optimization models for each.

```
budget=10 # Initial budget for January
# Looping to find the optimal allocation for all the months
for i in range(len(roi_by_month)):
    obj_month= np.array(return_by_month.iloc[i,:]) # objective vector
    A = np.zeros((13,10)) # initialize constraint matrix
    A[0,:] = [1,1,1,1,1,1,1,1,1,1] # budget constraint
    A[1,:] = [1,1,0,0,-1,0,0,0,0,-1] # print and TV constraint
    A[2,:] = [0,0,2,2,-1,-1,-1,-1,-1,0] # social media constraint
    A[3:13,:] = np.identity(10) # medium constraint

    # limits in M. Here the variable corresponding to the budget constraint (first) is kept variable
    # as the budegt keeps changing every month
    b = np.array([budget,0,0,3,3,3,3,3,3,3,3,3])

    sense = np.array(['<','<','<','<','<','<','<','<','<','<','<','<','<']) # all constraints are less than or equal constraints
    ROI_model_monthly= gp.Model() # initialize an empty model

    ROI_model_monthlyModX = ROI_model_monthly.addMVar(10) # tell the model how many variables there are

    # add the constraints to the model
    ROI_model_monthlyModCon = ROI_model_monthly.addMConstrs(A, ROI_model_monthlyModX, sense, b)

    ROI_model_monthly.setMObjective(None,obj_month,0,sense=gp.GRB.MAXIMIZE) # adding the objective to the model
    ROI_model_monthly.Params.OutputFlag = 0 # tell gurobi to shut up!!
    ROI_model_monthly.Params.TimeLimit = 3600
    ROI_model_monthly.optimize() # Optmizing the objective function
    obj_value_monthly=ROI_model_monthly.objVal # Storing the optimal objective for the month in loop
    allocation_platform_monthly=list(ROI_model_monthlyModX.x) # Storing the optimal allocation for the month in loop as a list
    allocation_platform_month_df.iloc[i]=allocation_platform_monthly # Adding the optimal allocation for the month in loop to df
    obj_value_month[month[i]]=obj_value_monthly # Adding the optimal objective for the month in loop to 'obj_value_month'
    return_monthly=(obj_value_monthly-budget)/budget # calculating monthly returns
    budget=budget+budget*return_monthly/2 # Updating budget by reinvesting half of the returns
```

Results:

Optimal allocation for different platforms for each of the months

	Print	TV	SEO	AdWords	Facebook	LinkedIn	Instagram	Snapchat	Twitter	Email
January	3.0	0.0	0.0	1.333333	0.0	0.0	2.666667	0.0	0.0	3.0
February	3.0	0.0	0.0	2.3955	3.0	0.0	0.0	0.0	1.791	0.0
March	0.0	0.0	0.0	3.0	0.0	3.0	1.389648	0.0	3.0	0.0
April	0.0	0.0	0.0	3.0	0.0	3.0	3.0	0.0	1.596856	0.0
May	1.8041	0.0	0.0	0.0	0.0	0.0	3.0	0.0	3.0	3.0
June	3.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	2.020172	3.0
July	1.123777	0.0	0.0	3.0	1.123777	0.0	3.0	0.0	3.0	0.0
August	3.0	0.0	0.0	1.827294	0.0	0.654588	0.0	0.0	3.0	3.0
September	1.362933	0.0	0.0	3.0	0.0	3.0	0.0	0.0	3.0	1.362933
October	0.0	0.0	0.0	3.0	0.0	3.0	3.0	0.0	0.0	2.955475
November	3.0	0.0	0.0	2.056421	0.0	1.112842	3.0	0.0	0.0	3.0
December	3.0	3.0	0.0	0.427951	3.0	0.0	0.0	0.0	0.0	3.0

Optimal Objective for each month

```
{'January': 10.373000000000001,
 'February': 10.592796,
 'March': 10.804064976,
 'April': 11.011343321567999,
 'May': 11.236243401451441,
 'June': 11.474938175936193,
 'July': 11.716209556193508,
 'August': 11.969848348460829,
 'September': 12.185085240357177,
 'October': 12.383050474650137,
 'November': 12.686638438510093,
 'December': 12.944784828665732}
```

Addressing stable budgets:

A stable budget is defined as a monthly allocation such that for each platform the monthly change in spend is no more than $1M. From our findings, we conclude that our allocation budget is not a stable budget as there are monthly allocation changes in spend of over $1M. One can model this by adding an additional constraint to which each marketing medium allocation change in spending is less than or equal to $1M.

CONCLUSION

This assignment relies heavily on linear programming to find an optimal solution for budget allocation across several marketing mediums with the objective of maximizing ROI returns. We used Gurobi to identify the optimal allocation of the marketing budget on the first set of ROI values. We then repeated this process on the second ROI dataset to find a new optimal budget allocation strategy. Together, we found the optimal ROI to be 4.56% and varying allocations across marketing mediums.

We then included a situation where we were given the opportunity to reinvest half of our returns and designed a new optimal allocation for each month, still abiding by the original constraints. From this, our results indicated that our budgets were not stable and that it would be sensible for us to run another optimization model with an added constraint spending changes.

In all, we conclude that our optimal budget allocation is dependent on accurate ROI estimates as the two allocation outputs we received were highly different. Given the information provided, there is not a way with certainty to conclude which option would be better for the company. If the values from the first test are correct, then it would be best to use that allocation and visa

versa. Overall, this assignment highlighted how helpful Gurobi can be in assessing business problems by finding optimal solutions that maximizes a given objective.