A picture containing text, font, graphics, logo

Description automatically generated

**BSAN 875: Advanced Business Analytics**

(Tuesday 7:05 to 9:45 pm)

Instructor: Murtaza Nazir

Spring 2023

Term Project Report (Team 11)

Optimizing Blending Aviation Gasoline at Jansen Gas using LP Models and Solver

**Team Members:**

Karthik Sai Twarakavi (V635P255)

Yogananda Theeguru (Q584C295)

Venkatasai Nallapati (F742K739)

Table of Contents

[INTRODUCTION 1](#_Toc134350593)

[OPTIMIZATION: 1](#_Toc134350594)

[Data Input 2](#_Toc134350595)

[Formulating Linear programming model: 3](#_Toc134350596)

[Defining decision variables: 3](#_Toc134350597)

[Defining constraints: 4](#_Toc134350598)

[Applying Gasoline constraints: 4](#_Toc134350599)

[Reid Vapor Pressure constraints: 5](#_Toc134350600)

[Octane Level constraints: 6](#_Toc134350601)

[Objective function: 7](#_Toc134350602)

[Problem Solving: 8](#_Toc134350603)

[Calculating the leftover feedstocks: 8](#_Toc134350604)

[Problem status and results: 10](#_Toc134350605)

[SENSITIVITY ANALYSIS 11](#_Toc134350606)

[RESULTS & CONCLUSION: 11](#_Toc134350607)

# INTRODUCTION

Jansen Gas is a Gasoline company which creates three types of aviation gases named A, B, C. To blend these three gasolines, it uses four different feedstocks which are Alkylate, Catalytic- Cracked Gasoline, Straight Run Gasoline, Isopentane.

Dave Wagner who is production manager of Jansen gas has collated data on available quantity of feed stocks where units of it are in gallons, value per gallon where units are in dollars, whereas it has chemical properties called Reid vapor pressure, and Octane ratings where it depends on the TEL which is Tetra Ethyl Lead which is used to increase the octane rating of gas. Which confirms that octane ratings and TEL levels are related to each other. Dave Wagner also compiled data on Gasolines where he assessed the required gasolines for the company and price at which they must be sold along with the assessed maximum Reid pressure that gasoline can have. Moreover, he calculated the minimum octane rating, TEL Level at which gasolines A, B, C must be blended. Below table would give a clear picture of the given data.

The main objective of the Jansen company is to achieve the optimal profit from selling the blended gases and leftover feedstocks so this can be done using Optimization.

# OPTIMIZATION:

It is a mathematical process of finding the best solution which could be either maximizing or minimizing for a given problem by considering constraints.

There are many types of Optimizations which are Linear programming, Integer programming and it depends on the type of problem. As the objective function is linear combination of decision variables with no quadratic terms, even the constraints on decision variables should linear moreover, decision variables take continuous values.

Every optimization problem involves the below process:

1. Data Input
2. Problem Definition
3. Choosing appropriate optimization technique.
4. Defining the Decision Variables
5. Defining the objective function
6. Building the constraints
7. Problem Solving
8. Checking the Status and printing the result.

## Data Input

Below is the data given by the Jansen gas company on Feed stocks and Gasolines where those were explained in the introduction.

A screenshot of a computer code

Description automatically generated with low confidence

Here we are defining the input variables in lists that we gathered from the problem statement.

## 

## Formulating Linear programming model:

A screen shot of a computer program

Description automatically generated with low confidence

Here we are defining our problem by the name “Gasoline Blending” inside ‘LpProblem ()’ function and as our goal is to maximize the revenue profits for the company, we are using ‘LpMaximize’ object from pulp library.

## Defining decision variables:

**Decision variables** are those in which we are control of. We can choose the values of these variables with which we can tune the result as we want.

A screenshot of a computer

Description automatically generated with low confidence

In these blocks of code, we are creating dictionary and defining the decision variables using ‘LpVariable.dicts ()’ function. We are naming this dictionary named as ‘x’.

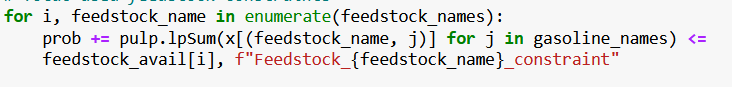
Using ‘for’ loops with variables ‘i’ and ‘j’ to iterate in feedstock\_names and gasoline\_names list we created a combination of these elements from these two lists which will be our decision variables. We set the lower bound as zero to ensure that there will not be any negative values and categories to be continuous. In the next line of code, we printed out those decision variables. As we use each feedstock in blending each gasoline and there are 4 feedstocks and 3 gasoline types. So, there are going to be 12 decision variables where each decision variable is the amount of feedstock used to make each gasoline.

## Defining constraints:

Constraints: In simple terms these can be defined as the requirements, or the limitations, or the conditions that we must follow or satisfy while finding a solution to a problem are called ‘Constraints’.

For this problem we have identified four types of constraints.

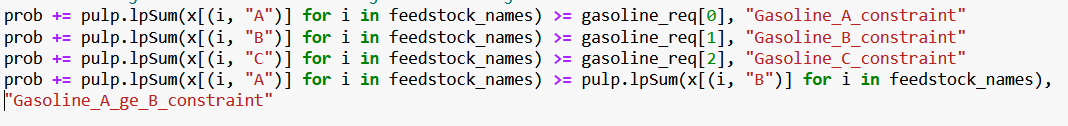
**Applying feedstocks constraints:**



The Jansen gas company gave us four feedstock constraints, one for each feedstock which should not exceed the gallons of each respective feedstock available. We have defined these gallon values in a list named ‘feedstock\_avail []’ above.

In the above line of code, we are defining this condition using a ‘for’ loop in which we used enumerate function to index each element in the collection (list in this case). Later on, on the L.H.S of the equation, we are adding the problem with sum all the linear expressions using ‘lpsum ()’ function which create the decision variables of feedstocks with each gasoline type. As it is maximum availability condition, we used ‘<=’ operator. On the R.H.S we gave the list that contains the values of maximum feedstock available.

## Applying Gasoline constraints:



Here we are defining the three constraints for each gasoline type and 1 for gasoline A.

We have a similar type of constraint for all three types of gasoline which is to produce a minimum required amount of or gallons of each gasoline type.

The first 3 lines of code do the same. To the existing prob we are adding the sum product of decision variables with each gasoline type on the L.H.S. Where are on the R.H.S we gave the ‘gasoline\_req[]’ which has the values of each gasoline type that has to be generated. We used ‘for’ loop and used ‘i’ as an iterative variable for ‘feedstock\_names[]’ list for each gasoline type A, B and C in each line respectively.

We also have a special constraint for gasoline A which is that gallons of A produced must be at least the gallons of B produced.

We define the same in the last line of code. In a comparable way to the above, on the L.H.S we defined the sum of all the decision variables of gasoline A and on the R.H.S we gave the sum of all the decision variables of gasoline B. We used ‘for’ loop and used ‘i’ as an iterative variable for ‘feedstock\_names[]’ list for gasoline type A.

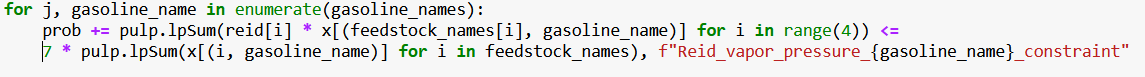
Since for all 4 constraints we must produce minimum specified gallons gasoline, we used ‘>=’ operator to satisfy the condition.

## Reid Vapor Pressure constraints:

A picture containing text, screenshot, number, font

Description automatically generated

Jansen gas company also gave us constraints for Reid Vapor Pressure for each feedstock and Max Reid Pressure for each gasoline type. In the following lines of code, we defined those constraints. Here we have little nested constraints which are we have constraints for both feedstocks and gasoline types. Since gasoline is made using all four feedstocks, we must satisfy the Reid Vapor Pressures of each feedstock should be less than each gasoline Reid Vapor Pressure constraints which in this case is same for all gasoline types unlike feedstocks.



We used ‘for’ loop to iterate in ‘gasoline\_names’ list with the iterative variable ‘i’. On the L.H.S we defined the sum of product of Reid Vapor Pressures with their respective feedstock. We gave for loop iterate in range (4) as we have 4 feedstocks. Whereas on the R.H.S we have the maximum Reid Vapor Pressure of each gasoline type is 7 This is multiplied by sum of feedstocks of each gasoline type. Since we have maximum values on the R.H.S we used ‘<=’ operator.

## Octane Level constraints:

**A picture containing text, font, screenshot

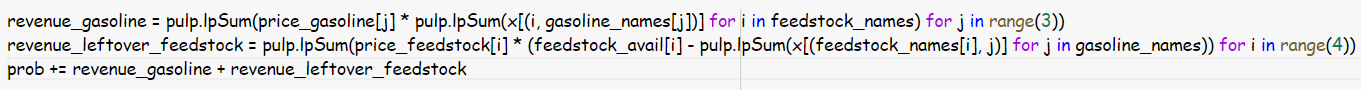
Description automatically generated**

Coming to octane level constraints, at first, we get the TEL level of each gasoline type which is ‘Low’ for gasoline A and ‘High’ for gasoline B and C. So, if we are making gasoline A since Low TEL level must be maintained the company gave us the Octane values for each feedstock and for gasoline types B and C since its High TEL level, we have Octane values for each feedstock. Also, we must maintain a minimum octane levels for each gasoline type.

It is the same logic we followed for all three octane level constraints. On L.H.S, we add the problem and the sum product of low TEL values with its respective feedstock for each gasoline type. We already gave the octane level values for low TEL and high TEL as lists at the beginning. It is low TEL for A and high TEL for B and C. Like earlier we gave an iterative variable to loop through a range of four for feedstocks and gasoline type is given manually. On R.H.S, we have the minimum octane value of each gasoline type multiplied by the sum of its respective feedstock, gasoline decision variable. Since its minimum value on the R.H.S we use ‘>=’ operator.

## Objective function:

In this part of the linear programming model, we define how to achieve our ultimate result. Simply put, we make a formula for the solution and define it in python here.



For Jansen Aviation Gasoline Blending company, we can obtain revenue from two ways which are by selling the gasoline made and by selling the leftover feedstocks. We are given the price values per gallon of gasoline and per gallon of leftover feedstocks.

In the first line, we calculate the revenue through gasoline. This is the product of price of each gasoline type with the gallons of gasoline produced. Gasoline produced is the sum product of all the decision variables of each gasoline type which we can get by iterating through the lists ‘gasoline\_names’ and ‘feedstock\_names’. To do this for gasoline types we have its range value up to 3 (0, 1, 2). For iterative variable of ‘feedstock\_names’ list since we have already used enumerate function at the time of constraints, we need not provide any range again. This is the primary reason for using the enumerate function. For each gasoline type we get 4 decision variables with each feedstock at a time.

In the second line of code, we first calculate the amount of feedstock left by subtracting the used feedstock from the available feedstock. And then we multiply it with the price per gallon value of each feedstock.

E.g., Let us consider feedstock Alkylate. We sum up all the decision variables with this feedstock which are (Alkylate, A), (Alkylate, B) and (Alkylate, C) and subtract this sum value from the feedstock availability of feedstock Alkylate from the ‘feedstock\_avail[]’ list and multiply this with price per gallon value of Alkylate leftover.

With the range function and value of 4, we go through all the feedstocks in the list each time.

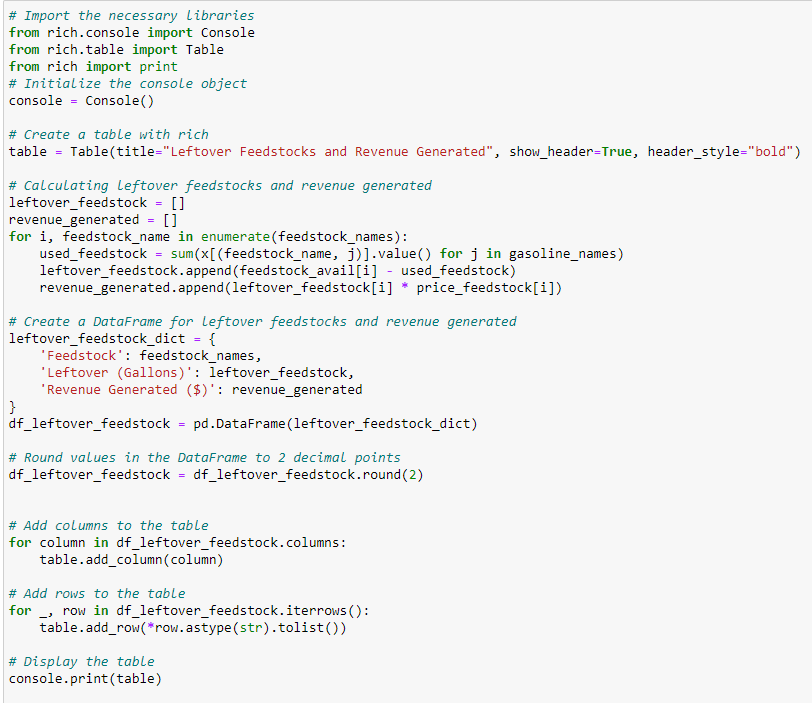
In the last line of code, we add the above two revenues to get our total final revenue.

## Problem Solving:

****

Here we initiate the solve () function to start solving the problem.

## Calculating the leftover feedstocks:



In this piece of code, we calculated the gallons of leftover feedstocks and revenue generated by selling. Rich is a python library where we can create styled console and display the formatted text. We have imported ‘table’ class from rich.table to create and format table and print function from rich library to print the styled content. Also imported the ‘Console’ class from ‘rich.console’ module to print the table to console. console = Console (): Create a Console instance that you will use to print the table. we created table titled **Leftover Feedstocks and Revenue Generated** with header and created two empty list for left over feed stocks and revenue generated from it to store the values and used for loop to iterate through names of feedstock using their index and name whereas used\_feedstock calculates the total amount of feedstock used by summing the values of decision variables x for each feedstock and gasoline type and the difference between the available feedstock and used feedstock is appended to the leftover\_feedstock list later the revenue generated from the leftover feedstock is calculated by multiplying the leftover amount with its price and appended to the revenue\_generated list.

leftover\_feedstock\_dict is a dictionary containing the feedstock names, leftover amounts, and revenue generated. df\_leftover\_feedstock is a Data Frame created from the leftover\_feedstock\_dict. In the end the values in the Data Frame df\_leftover\_feedstock is rounded to 2 decimal points. Then added columns and rows with their values using for loop. **for \_, row in df\_leftover\_feedstock.iterrows():** iterates over the rows of the Data Frame, with row representing the data of the current row. ‘\_’ used as a placeholder for the row index. **row.astype(str).tolist():** This expression first converts the data type of each element in the row to a string using row.astype(str) and then converts the row to a list using tolist(). The list will contain the string representations of each element in the row. Finally prints the formatted table to console.

## Problem status and results:

**A screenshot of a computer program

Description automatically generated with medium confidence**

After initiating the solve function above, we find out its status if its ‘optimal’ or ‘infeasible’. If it is optimal, we print it out as Optimal solution found which we did. Else, we print no optimal solution found.

To print out the decision variables and the optimal values we got after solving we created a dictionary with Feedstock, Gasoline and Gallons used as keys lists. Later we append the names of feedstocks and gasoline types which are defined at the beginning of the code. Whereas for the values or here gallons of feedstocks of each type of gasoline, we append those decision variables like we defined earlier along with ‘value ()’ function which fills the values we got in the solution.

Later we converted the dictionary into a data frame and printed the data frame. Then we have created the table instance by naming the title as “Decision variables” and set the headers as true which means instructing it to show and to be displayed in bold. Next line of code “**for column in df\_decision\_variables.columns:”** iterates through the columns in data frame and adds each column to table using “**table.add\_column(column)”.** Next few lines of code iterate over the rows and convert the row’s data to string and add it to table. We print the table using console instance. Finally print the total revenue which is our objective function.

# Sensitivity Analysis:

A screen shot of a computer code

Description automatically generated with low confidence

The above code helps in printing the shadow price and slack for the constraints of the problem. Firstly, we are creating list of dictionaries where each dictionary has slack, shadow price, slack value of defined constraint and then we convert it to a Data Frame and using table object from the ‘rich’ library we display the headers in bold style by keeping title as sensitivity analysis of constraints. Used for which iterates over the columns and rows and with help of ‘table’ function we add values table. Using **table.add\_row(\*row.astype(str).tolist())** with values from current row adds new row and converting them to strings and unpacking the list as separate arguments. In the prints the formatted table with display of constraints, slack values, and shadow prices.

# Results & Conclusion:

Units of Feedstocks used for blending gasolines:

*Decision Variables*

┏━━━━━━━━━━━━┳━━━━━━━━━━┳━━━━━━━━━━━━━━┓

┃ **Feedstock** ┃ **Gasoline** ┃ **Gallons Used** ┃

┡━━━━━━━━━━━━╇━━━━━━━━━━╇━━━━━━━━━━━━━━┩

│ Alkylate │ A │ 37131.064 │

│ Alkylate │ B │ 34312.268 │

│ Alkylate │ C │ 68556.668 │

│ CCG │ A │ 87887.023 │

│ CCG │ B │ 0.0 │

│ CCG │ C │ 0.0 │

│ SRG │ A │ 4899.36 │

│ SRG │ B │ 73457.249 │

│ SRG │ C │ 61643.391 │

│ Isopentane │ A │ 82.552639 │

│ Isopentane │ B │ 22230.483 │

│ Isopentane │ C │ 24772.578 │

└────────────┴──────────┴──────────────┘

Above table explains the total amount of feedstock that has been used in blending the gasolines A, B, C. We can deduce that CCG is not used in blending gasolines B, C.

**Left over feedstocks:**

*Leftover Feedstocks and Revenue Generated*

┏━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━┓

┃ **Feedstock** ┃ **Leftover (Gallons)** ┃ **Revenue Generated ($)** ┃

┡━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━┩

│ Alkylate │ 0.0 │ 0.0 │

│ CCG │ 42112.98 │ 105282.44 │

│ SRG │ 0.0 │ 0.0 │

│ Isopentane │ 62914.39 │ 147848.81 │

└────────────┴────────────────────┴───────────────────────┘

The above table gives the result of gallons of feedstocks which are unutilized for blending the gases.

**Sensitivity Analysis:**

We have calculated shadow price and slack, where shadow price is it observes the affect of revenue variable by change in one unit of the constraint. Slack is the difference between left hand side and right-hand side of a constraint. Below are the calculated slack and shadow prices of the defined constraints.

*Constraints*

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━┓

┃ **name** ┃ **shadow price** ┃ **slack** ┃

┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━┩

│ Feedstock\_Alkalyte\_constraint │ 1.2665105 │ -0.0 │

│ Feedstock\_CCG\_constraint │ -0.0 │ 42112.977 │

│ Feedstock\_SRG\_constraint │ 0.44846025 │ -0.0 │

│ Feedstock\_Isopentane\_constraint │ -0.0 │ 62914.386 │

│ Gasoline\_A\_constraint │ -0.0 │ -10000.0 │

│ Gasoline\_B\_constraint │ -0.38903163 │ -0.0 │

│ Gasoline\_C\_constraint │ -0.0 │ -34972.640000000014 │

│ Gasoline\_A\_ge\_B\_constraint │ -0.44037659 │ -0.0 │

│ Reid\_vapor\_pressure\_A\_constraint │ 0.27230544 │ -0.0 │

│ Reid\_vapor\_pressure\_B\_constraint │ 0.24001948 │ -0.0 │

│ Reid\_vapor\_pressure\_C\_constraint │ 0.24001948 │ -0.0 │

│ Octane\_A\_constraint │ -0.22269038 │ -0.0 │

│ Octane\_B\_constraint │ -0.18378165 │ 7.9092892e-12 │

│ Octane\_C\_constraint │ -0.18378165 │ 7.1636916e-12 │

└──────────────────────────────────┴──────────────┴─────────────────────┘

Shadow price: From the above table we can deduce that increase in one gallon of usage of Alkylate feed stock to blend gases would increase the revenue by $1.26 and increase in one gallon usage of CCG constraint would not have any effect on the revenue thus it is 0, whereas one gallon increased usage of SRG would increase the revenue by $0.44 of revenue. Same approach for Isopentane.

Coming to gasoline, if an additional one gallon of gasoline A is produced this will not increase the revenue of Jansen gas, similarly for gasoline C too, whereas additional blend of gasoline B will decrease the revenue by $0.38.

For the Reid vapor pressure constraints, one unit increase of RVP of gas A would increase the revenue by $0.27 and similar for other two RVP constraints.

If one unit increase of octane rating of gas A would reduce the revenue by $0.22 and similarly for other two constraints.

Slack: It is an unused portion of resource at optimal solution.

In feedstock constraints negative slack indicates that all the resources have been utilized as constraint is we can not use more than certain portions of feedstocks to blend gases. Positive slack of 42122.977 indicates that 42122.977 gallons of CCG feedstock was not utilized for blending gasolines and similarly for other feedstocks.

Negative slack in gasoline constraint indicates that additional number of gasolines were produced than asked by the Jansen gas, for example -10000 for gasoline A is nothing but additional thousand gallons of Gasoline A is produced than required gasoline same way it applies for gasoline C too. 0 for gasoline B is exact amount of gas B is blended as asked by Jansen gas.

RVP constraints are neither positive nor negative as exact units of RVP must be maintained for blending it should not be increased or decreased.

For octane rating for gas A constraint, it is zero, whereas for B & C gases it is 7.99e-12 and 7.16e-12 which indicates additional units of octane ratings were utilized, for this problem it can not be negative as minimum octane ratings must be taken for blending the gases.

Revenue: After selling the blended gases and leftover feedstocks for the mentioned price, Jansen gas achieved revenue of **$1,718,021.80**.