Individual Report

1. Introduction

In this paper, I used Zero-One Integer Programming to solve the problem of selecting the optimal skincare product mix to maximize revenue. The problem definition includes considerations of product type, brand, skin type suitability, and expected customer budget constraints. Finally, I analyzed this integer programming model's advantages and disadvantages and proposed a direction for improvement.

1.1 Problem Background

As a product strategist at LOOKFANTASTIC, an online premium beauty retailer, my main responsibility is to curate an optimal selection of skincare products for the upcoming Boxing Day sale. The objective is to cater to the diverse needs of our target audience while maximizing total revenue for the company. To achieve this goal, I propose selecting nine product sets specifically designed for different skin types (oily, combination, and normal/dry) and budget limitations (under £30, under £50, and under £100).

1.2 Data Collection

```
In [189]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1138 entries, 0 to 1137
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   product_name   1138 non-null   object
 1   product_url    1138 non-null   object
 2   product_type   1138 non-null   object
 3   clean_ingreds  1138 non-null   object
 4   price          1138 non-null   object
dtypes: object(5)
memory usage: 44.6+ KB
```

Figure 1 – The Information of Dataset

The primary dataset of facial skincare products in LOOKFANTASTIC was collected from

the Kaggle notebook, Skincare Recommendation Engine. As shown in Figure, it comprises five columns, of which product_name, product_type and price were used to establish a linear programming model for facial skincare product portfolios, reducing the dataset to 646 relevant products after excluding non-facial items like Bath Salts and Body Wash.

To estimate expected customer numbers, I referred to SimilarWeb data, which indicated that LOOKFANTASTIC had 10.1 million visits in December 2023, with approximately 78.93% from the UK. Furthermore, according to Barclays Bank's report (Barclays Bank 2023), total spending on Boxing Day in 2023 reached £4,673,118,955.20, with 23% of consumers choosing to buy beauty items. Considering the beauty industry's typical 1% to 3% conversion rate and seasonal data collection constraints, a conservative 3% conversion rate was applied, projecting 1,774 potential customers, as detailed in Figure 2.

```
# Percentage of skincare products in total Boxing Day spending
boxing_day_skincare_percentage = 23 / 100

# Total Boxing Day spending
total_boxing_day_spend_gbp = 4673118955.20

# Total Boxing Day spending on skincare products
boxing_day_skincare_spend_gbp = total_boxing_day_spend_gbp * boxing_day_skincare_percentage

# Percentage of UK visits in Lookfantastic's total December visits
uk_visits_december = 10.1e6 * (78.93 / 100)

# Conversion rate of 3%
conversion_rate = 0.03

# Estimating the number of customers expected to make a purchase
estimated_buyers = uk_visits_december * conversion_rate

# Average spending per buyer on skincare products
average_spend_per_buyer_gbp = boxing_day_skincare_spend_gbp / estimated_buyers

# Estimating the number of customers per month
estimated_monthly_customers = uk_visits_december / average_spend_per_buyer_gbp

# Output: Estimated number of monthly customers
estimated_monthly_customers = 1773.84
```

Figure 2 – Calculation of Estimated Potential Customers

2. Decision Making Framework

2.1 Objective Function

$$\text{Objective Function} = 1774 \times 0.159 \times (\sum \text{Price of Oily Skin Gift Sets}) + 1774 \times 0.356 \times (\sum \text{Price of Combination Skin Gift Sets}) + 1774 \times 0.485 \times (\sum \text{Price of Normal/Dry Skin Gift Sets})$$

Based on the survey by Fawkes et al. (2021), Britons' skin types can be divided into oily (15.9%), combination (35.6%), and normal/dry (48.5%).

Besides, I assume the revenue:

(1) Even distribution of skin types across budget ranges: Customers of each skin type are evenly distributed across different budget ranges, resulting in an equal representation of each skin type in each budget range.

(2) Independent customer decision-making: The decision of a customer to choose a product set within a budget range does not affect the probability of them choosing product sets from other budget ranges.

2.2 Decision Variables

In the context of Zero-One Integer Programming, each of the 646 products in our dataset is represented by a decision variable. These variables are binary, meaning they can take on a value of either 1 or 0. A value of 1 indicates that the corresponding product is selected to be part of the optimal product set, while a value of 0 indicates that the product is not selected. This binary decision framework allows us to identify the best combination of products that maximizes our objective function—the total revenue—while adhering to constraints like budget limits and customer skin type preferences.

2.3 Constraints

I implemented the following constraints in the model to increase the product diversity of different sets.

Skin Type	Price Range	Required Products
Normal/Dry	Under £30	Toner, Serum
Combination		Toner, Serum
Oily		Mist, Serum
Normal/Dry	Under £50	Toner, Serum, Moisturiser
Combination		Toner, Serum, Peel
Oily		Mist, Serum, Exfoliator
Normal/Dry	Under £100	Toner, Serum, Moisturiser, Exfoliator, Oil
Combination		Toner, Serum, Peel, Exfoliator, Eye Care
Oily		Mist, Serum, Exfoliator, Peel, Eye Care

Table 1 - Skin Care Product Sets by Skin Type and Price Range

(1) Table 1 organizes the requirements based on skin type and price range, listing the products that must be included in each set.

(2) Within each product set, the brands of the products must not repeat.

(3) Across the 9 types of product sets, there must be no repeated products; for example, the Mist in the under £30 gift box for oily skin cannot be the same as the Mist in the £50 pounds gift box for oily skin.

I've drawn my own conclusions on the must-have products for different skin types by referring to all the major online forums, including Reddit and Twitter, PTT and Dcard in Taiwan, where users with different skin types have shared their thoughts on their favourite products.

2.4 Model Building

The model is built using python with Gurobi, where the syntax of model building includes defining decision variables, objective functions and setting constraints, etc. For the complete model syntax, please refer to the Appendix.

3. Results

Through Gurobi, we were able to obtain an optimal solution report. The report indicates that the optimal solution is equal to the best bound, meaning an exact optimal solution has

been found. The gap between the solutions is 0%, which means that the solution is globally optimal. The optimal objective value is £319,320.

```
Optimize a model with 2068 rows, 5517 columns and 26962 nonzeros
Model fingerprint: 0xd6de14d3
Variable types: 0 continuous, 5517 integer (5517 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+02]
  Objective range  [7e+02, 2e+05]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+02]
Found heuristic solution: objective 311500.95308
Presolve removed 868 rows and 1542 columns
Presolve time: 0.11s
Presolved: 1200 rows, 3975 columns, 23841 nonzeros
Variable types: 0 continuous, 3975 integer (3975 binary)
Found heuristic solution: objective 318507.86280

Root relaxation: objective 3.193200e+05, 1119 iterations, 0.03 seconds (0.02 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 319320.000    0   18 318507.863 319320.000  0.25%     -    0s
H    0     0                     319083.02908 319320.000  0.07%     -    0s
...
Solution count 4: 319320 319083 318508 311501

Optimal solution found (tolerance 1.00e-04)
Best objective 3.193200000000e+05, best bound 3.193200000000e+05, gap 0.0000%
```

Figure 3 – The Optimal Solution Report

Moreover, I have also used a 3X3 matrix to show the results for each set of the best solutions.

	≦ £30	≦ £50	≦ £100
Oily	1. The Ordinary Marine Hyaluronics 30ml 2. Neutrogena Hydro Boost Express Hydrating Spray 200ml 3. AHAVA Single Use Facial Renewal Peel 8ml 4. Sukin Purely Ageless Eye Cream 20ml	1. Avene Hydrance Intense Serum 30ml 2. Lumene Nordic Hydra [Lähde] Arctic Spring Water Enriched Facial Mist 100ml 3. Mama Mio The Tummy Rub Scrub 180ml	1. Murad Retinol Youth Renewal Serum Travel Size 2. Revolution Skincare Superfruit Essence Spray 100ml 3. Mavala Skin Vitality Beauty Enhancing Micro-Peel 65ml 4. La Roche-Posay Redermic [R] Retinol Eye Cream 15ml 5. Polaar Arctic Cotton Gentle Scrub 50ml
Price/Each Set	£30	£50	£100
Normal/Dry	1. NIOD Multi-Molecular Hyaluronic Complex 15ml 2. La Roche-Posay Serozinc Toner 50ml	1. First Aid Beauty Coconut Skin Smoothie Priming Moisturiser 2. La Roche-Posay Rosaliac AR Intense Serum 40ml 3. Sanctuary Spa Daily Glow Radiance Tonic 150ml	1. La Roche-Posay Toleriane Sensitive Moisturiser 40ml 2. Holika Holika 3 Seconds Starter (Collagen) 3. DHC Deep Cleansing Oil 70ml 4. The Ordinary Lactic Acid 10% + HA 2% Superficial Peeling Formulation 30ml 5. Sukin Hydrating Mist Toner (125ml) 6. Filorga Scrub and Mask 55ml
Price/Each Set	£30	£50	£100
Combination	1. Sanctuary Spa Hyaluronic Wonder Oil Serum 30ml 2. PIXI Rose Tonic 100ml	1. The Ordinary Hyaluronic Acid 2% + B5 Supersize Serum 60ml 2. Garnier Ambre Solaire Natural Bronzer Quick Drying Dark Self Tan Face Mist 75ml 3. Embryolisse Gentle Night Peeling Care Radiance Secret 40ml 4. Liz Earle Instant Boost Skin Tonic 200ml Bottle	1. The Ordinary Buffet Supersize Serum 60ml 2. Elizabeth Arden Visible Difference Peel & Reveal Revitalizing Mask (50ml) 3. Avene Gentle Eye Make-Up Remover 125ml 4. Lancôme Tonique Confort Toner 200ml 5. Gallinée Prebiotic Face Mask and Scrub 100ml
Price/Each Set	£30	£50	£100

Table 2 – The Optimal Product Sets

4. Discussion

Based on Table 2, the optimal solution demonstrates that our gift box not only caters to a variety of skin textures but also features a diverse array of brands. This strategy enables customers to explore and familiarize themselves with a broad selection of brands. Additionally, by strategically curating gift boxes that avoid product overlap, we effectively incentivize customers to explore broader options, potentially leading to higher sales across multiple price segments.

To increase sales, the following recommendations can be made to the product mix to match the actual market situation and reflect the diversified customer needs.

4.1 Recommendations

(1) Product Diversification: To cater to a wider audience, consider incorporating more

skin types and sensitivities into the product selection criteria. This could involve adding more specific categories like sensitive skin, aging skin, or acne-prone skin, thereby appealing to a broader customer base.

(2) Seasonal Adjustment: The current product mix overlooks the significant variations in seasonal skincare needs. Incorporating seasonal variations into the product mix can attract customers looking for solutions to seasonal skin problems, such as dry skin in winter or sun protection in summer.

(3) Customer Engagement and Feedback Analysis: The cosmetics marketplaces a lot of emphasis on word-of-mouth marketing because most customers' purchasing behaviour is influenced by what others say about a product. Therefore, besides offering various products for different skin types and budget ranges, it is essential to consider how to incorporate customer engagement and feedback into the product portfolio. For example, analysing customer reviews and feedback on products sold in the past can provide insights into consumer preferences and product performance, leading to the design of more appealing gift boxes.

5. Improvements

The decision model choice, while effective, relies on several assumptions about customer distribution and decision-making that may not fully capture real-world complexities. For instance, assuming an even distribution of skin types across budget ranges might oversimplify consumer behavior, as certain skin types might have a predisposition towards higher or lower spending. Moreover, the model does not consider external factors such as competitor actions, market trends, or economic changes, which can significantly influence consumer spending behavior.

In the following, I will suggest ways to improve the model and make it more comprehensive.

(1) Consider the revenue within each budget range separately: For each skin type,

calculate the revenue based on the expected number of customers and the anticipated distribution ratio across different budget ranges. Then, aggregate these revenues to arrive at the total income.

(2) Incorporate Purchase Preferences: If data is available to collect, adjust the model to reflect known customer preferences for certain types of products within different budget ranges. This adjustment could lead to a more targeted and effective product selection.

(3) Incorporating Costs and Discounts: The current decision model does not fully account for the real income after discounts due to the lack of direct access to product costs and the company's discount strategies. To enhance the model's accuracy and applicability, future improvements should aim to integrate detailed cost data and discount policies. This integration would enable a more comprehensive evaluation of revenue, reflecting the actual financial performance more accurately.

(4) Expansion to New Markets: Because customers in LOOKFANTASTIC are all over the world, we can consider the potential for expanding the product mix strategy to new geographical markets. Different regions may have different skin care preferences and budgetary constraints, requiring adjustments to the model.

6. References

[1] *Shoppers to spend £253 in festive sales, the highest amount in four years | Barclays* (no date). https://home.barclays/news/press-releases/2023/12/shoppers-to-spend-p253-in-festive-sales--the-highest-amount-in-f/.

[2] Eward (2021) 🌸 *Skincare Recommendation Engine* 🌸. https://www.kaggle.com/code/eward96/skincare-recommendation-engine/notebook.

[3] Similarweb (2024) *lookfantastic.com Ranking by Traffic.* https://www.similarweb.com/website/lookfantastic.com/#geography

[4] Mirrors, H. (no date) *The Price of Beauty in the UK 2024 [Infographics]*. https://www.hollywoodmirrors.co.uk/blogs/news/the-price-of-beauty-uk-2022-infographics.

[5] Fawkes, N. *et al.* (2021) 'A survey to identify determinants that Influence Self-PErceived Sensitive Skin in a British population: Clues to developing a Reliable Screening Tool for Sensitive skin,' *Clinical, Cosmetic and Investigational Dermatology*, Volume 14, pp. 1201–1210. https://doi.org/10.2147/ccid.s317970.

7. Appendix

```
!pip install gurobipy
```

```
Requirement already satisfied: gurobipy in c:\users\ruby cheng\anaconda3\lib\site-packages (11.0.0)
```

```
from gurobipy import Model, GRB, quicksum
import pandas as pd

# Load data
df = pd.read_excel('dataset.xlsx')
# Remove currency symbol and convert price to float
df['price'] = df['price'].str.replace('£', '').astype(float)
# Exclude products containing "SPF" in their names (case insensitive)
df = df[~df['product_name'].str.contains("SPF", case=False)]

# Create optimization model
model = Model("Maximize_Revenue")

# Define skin types, budget ranges, and products
skins = ['oily', 'normal_dry', 'combination']
budgets = ['under_30', 'under_50', 'under_100']
products = df['product_code'].unique().tolist()

# Add decision variables for selecting products within skin types and budget ranges
decision_variables = model.addVars(products, skins, budgets, vtype=GRB.BINARY, name="decision_variables")

# Percentages of customer base by skin type
skin_type_percentages = {"oily": 0.159, "normal_dry": 0.485, "combination": 0.356}
```

```
# Estimated number of customers
estimated_customers = 1774

# Set objective function to maximize total revenue
# Revenue is calculated as the sum of product prices times the number of customers
# times the percentage of customers with each skin type, for selected products across all skin types and budget ranges
model.setObjective(
    quicksum(
        decision_variables[p, s, b] * df.loc[df['product_code'] == p, 'price'].values[0] * estimated_customers * skin_type_percentages[s]
        for p in products
        for s in skins
        for b in budgets
    ),
    GRB.MAXIMIZE
)
```

```
Set parameter Username
Academic license - for non-commercial use only - expires 2025-01-25
```

```
# Required product types for each skin type and budget range
required_products_by_skin_budget = {
    ('normal_dry', 'under_30'): ['Toner', 'Serum'],
    ('combination', 'under_30'): ['Toner', 'Serum'],
    ('oily', 'under_30'): ['Mist', 'Serum'],
    ('normal_dry', 'under_50'): ['Toner', 'Serum', 'Moisturiser'],
    ('combination', 'under_50'): ['Toner', 'Serum', 'Peel'],
    ('oily', 'under_50'): ['Mist', 'Serum', 'Exfoliator'],
    ('normal_dry', 'under_100'): ['Toner', 'Serum', 'Moisturiser', 'Exfoliator', 'Oil'],
    ('combination', 'under_100'): ['Toner', 'Serum', 'Peel', 'Exfoliator', 'Eye Care'],
    ('oily', 'under_100'): ['Mist', 'Serum', 'Exfoliator', 'Peel', 'Eye Care']
}

# Adding constraints for required product types
for (skin, budget), product_types in required_products_by_skin_budget.items():
    for product_type in product_types:
        products_of_type = df[df['product_type'] == product_type]['product_code'].tolist()
        model.addConstr(quicksum(decision_variables[product, skin, budget] for product in products_of_type) >= 1, name=f"constraint_{skin}_{budget}_
```

```python
# Adding budget constraints
budget_limits = {
    'under_30': 30,
    'under_50': 50,
    'under_100': 100
}

for skin in skins:
    for budget in budgets:
        total_cost = quicksum(df.loc[df['product_code'] == p, 'price'].values[0] * decision_variables[p, skin, budget] for p in products)
        model.addConstr(total_cost <= budget_limits[budget], name=f"Budget_limit_{skin}_{budget}")
```

```python
# Adding constraints for brand uniqueness
brands = df['brand'].unique().tolist()

for skin in skins:
    for budget in budgets:
        for brand in brands:
            products_of_brand = df[df['brand'] == brand]['product_code'].tolist()
            model.addConstr(quicksum(decision_variables[product, skin, budget] for product in products_of_brand if product in products) <= 1, name=f
```

```python
# Adding constraints for product type uniqueness
product_types = df['product_type'].unique().tolist()

for skin in skins:
    for budget in budgets:
        for product_type in product_types:
            products_of_type = df[df['product_type'] == product_type]['product_code'].tolist()
            model.addConstr(quicksum(decision_variables[product, skin, budget] for product in products_of_type if product in products) <= 1, name=f'
```

```python
# Ensuring each product is selected at most once across all combinations
for product in products:
    model.addConstr(sum(decision_variables[product, skin, budget] for skin in skins for budget in budgets) <= 1, name=f"Product_{product}_unique_acr
```

```python
# Solve the problem
model.optimize()
```

```
Gurobi Optimizer version 11.0.0 build v11.0.0rc2 (win64 - Windows 11+.0 (22631.2))

CPU model: 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz, instruction set [SSE2|AVX|AVX2|AVX512]
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 2068 rows, 5517 columns and 26962 nonzeros
Model fingerprint: 0xd6de14d3
Variable types: 0 continuous, 5517 integer (5517 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+02]
  Objective range  [7e+02, 2e+05]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+02]
Found heuristic solution: objective 311500.95308
Presolve removed 868 rows and 1542 columns
Presolve time: 0.11s
Presolved: 1200 rows, 3975 columns, 23841 nonzeros
Variable types: 0 continuous, 3975 integer (3975 binary)
Found heuristic solution: objective 318507.86280
```

```
Root relaxation: objective 3.193200e+05, 1119 iterations, 0.03 seconds (0.02 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 319320.000    0   18 318507.863 319320.000  0.25%     -    0s
H    0     0                     319083.02908 319320.000  0.07%     -    0s
...
Solution count 4: 319320 319083 318508 311501

Optimal solution found (tolerance 1.00e-04)
Best objective 3.193200000000e+05, best bound 3.193200000000e+05, gap 0.0000%
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```python
    # print the result
    for v in model.getVars():
        if v.x > 0:
            print(f"{v.varName} = {v.x}")
```

```
decision_variables[63,normal_dry,under_100] = 1.0
decision_variables[95,normal_dry,under_50] = 1.0
decision_variables[116,combination,under_100] = 1.0
decision_variables[121,combination,under_50] = 1.0
decision_variables[122,oily,under_30] = 1.0
decision_variables[123,oily,under_50] = 1.0
decision_variables[132,normal_dry,under_30] = 1.0
decision_variables[138,normal_dry,under_50] = 1.0
decision_variables[148,combination,under_30] = 1.0
decision_variables[183,oily,under_100] = 1.0
decision_variables[200,normal_dry,under_100] = 1.0
decision_variables[229,normal_dry,under_100] = 1.0
decision_variables[308,oily,under_30] = 1.0
decision_variables[310,combination,under_50] = 1.0
decision_variables[322,oily,under_50] = 1.0
decision_variables[352,oily,under_100] = 1.0
decision_variables[569,normal_dry,under_100] = 1.0
decision_variables[573,oily,under_30] = 1.0
decision_variables[575,combination,under_100] = 1.0
decision_variables[584,combination,under_50] = 1.0
```

```python
# Initialize an empty DataFrame as a matrix
matrix_df = pd.DataFrame(index=skins, columns=budgets, dtype='object')

# Initialize an empty list for each cell in the DataFrame
for skin in skins:
    for budget in budgets:
        matrix_df.at[skin, budget] = []

# Iterate through decision variables, if a product is selected, then add it to the respective cell
for v in model.getVars():
    if v.x > 0:  # If the product is selected
        # Parse product_code, skin, budget from the variable name
        details = v.varName.split('[')[1].split(']')[0].split(',')
        product_code = details[0]
        skin = details[1].strip().strip('"')
        budget = details[2].strip().strip('"')

        # Add product_code to the corresponding cell
        matrix_df.at[skin, budget].append(product_code)

# Convert the list in each cell to a string for easy printing and viewing
for skin in skins:
    for budget in budgets:
        matrix_df.at[skin, budget] = ', '.join(matrix_df.at[skin, budget])
```

```python
    # Print the result
    print(matrix_df)
```

```
                      under_30                under_50  \
oily         122, 308, 573, 679      123, 322, 917
normal_dry                132, 819       95, 138, 850
combination               148, 823  121, 310, 584, 821

                                 under_100
oily                183, 352, 595, 613, 938
normal_dry   63, 200, 229, 569, 830, 925
combination         116, 575, 636, 840, 923
```

```
total_revenue = 0

for v in model.getVars():
    if v.x > 0:  # Only consider selected products
        # Parse product_code, skin_type from the variable name
        parts = v.varName.split('[')[1].split(']')[0].split(',')
        product_code = parts[0].strip().strip('"')
        skin_type = parts[1].strip().strip('"')

        # Find the corresponding price
        matched_prices = df.loc[df['product_code'].astype(str) == str(product_code), 'price'].values

        if len(matched_prices) > 0:
            price = matched_prices[0]
            # Calculate the contribution of this product to total revenue
            revenue_contribution = price * v.x * estimated_customers * skin_type_percentages[skin_type]
            total_revenue += revenue_contribution
        else:
            print(f"Price not found for product_code: {product_code}")

print(f"Total Revenue: £{total_revenue:.2f}")
```

```
Total Revenue: £319320.00
```