University of Waterloo

Department of Management Sciences

MSCI 433: Applications of Management Engineering (Winter 2024)

Term Project Part 2

Decoding Bank Customer Churn

Prepared for:

Prof. Fatma Gzara

Prepared by Group 18:

Chen, Yali (20831303)

Liaw, Cuthbert (20846102)

Nonta, Nayeema (20837920)

Date Submitted:

April 11th, 2024

Abstract

This report addresses the critical issue of customer churn in the banking sector. With the backdrop of increased competition and fluctuating interest rates, banks face the challenge of retaining customers while optimizing promotional spending. The report introduces a dual-model approach that uses advanced prescriptive analytics to proactively minimize customer churn by tailoring promotional strategies to individual customer risk profiles.

The initial part of the report introduces categorization of customers based on their sensitivity to interest rate changes. This distinction is pivotal in developing targeted interventions that are more likely to succeed than generic promotions. Employing logistic regression techniques with Python's Scikit-Learn, the team predicts churn probabilities, which then feed into optimization models designed to allocate promotions effectively under budget constraints.

Two distinct models are discussed: Model 1, a basic single-tier promotion strategy, and Model 2, a more complex multi-tier approach that incorporates a trade-off parameter allowing a balance between minimizing churn and controlling promotional costs. Both models utilize Gurobi and PuLP for optimization, ensuring that promotions are distributed in a way that maximizes retention while adhering to budgetary limitations.

The report presents results from sensitivity analyses that assess the impact of various budget levels and trade-off parameters on the effectiveness of these promotional strategies. The results are visually supported by Matplotlib-generated graphs, providing clear insights into the effectiveness of different promotional allocations.

In conclusion, the report underscores the effectiveness of using prescriptive analytics in formulating strategic decisions in the banking industry. By leveraging prescriptive modeling to assess churn risk and applying sophisticated optimization techniques, banks can more accurately target promotions to customers most at risk of defection, improving financial performance.

1. Problem Statement

High customer churn rate poses significant challenges for banks. With the rising rate environment, and increased competition from digital banks, bank churn has become a pressing issue (Marous, 2023). This leads to the problem statement explored in this report, how can banks proactively anticipate customer churn risks, and deploy appropriate countermeasures to different consumer groups in ensuring deposit retention with an efficient budgetary use.

2. Prescriptive Problem

Customers in a bank can be generally segregated into rate sensitive and non-rate sensitive customers. In hindsight, the differences in these customers are in how they behave when interest rates shift due to economic conditions. Rate sensitive customers always to look to maximize the interest they receive and would hesitate less in moving to competitors should they get better rates.

In terms of retention strategies, banks typically offer either promotional interest rates or any direct cash (i.e., deposit bonus, gifts, etc.) to clients they deem are of high-risk. The subsequent prescriptive problem would be on how to use churning probabilities to minimize total churning. As seen on model 1, we will perform this by ensuring appropriate promotion distribution to customers most at risk of churning. Model 2 will incorporate segregated promotional tiers, and trade-off parameters between increasing promotion budget and minimizing churn rate.

3. Prescriptive Tools

Our prescriptive solution implementation is centered on utilizing the outcomes of our predictive analysis to optimize the distribution of promotions effectively. To implement our solution we used the tools as detailed below.

3.1. Predicting Churn Probabilities using Logistic Regression and Scikit-Learn

Firstly, to get the churn probabilities of each customer, we leveraged logistic regression model from Scikit-Learn, with testing and training set derived from Kaggle. The logistic regression model allows us to receive a value between 0 and 1, indicating churning probability for each customer.

3.2. Data Management with Pandas, Optimization with Gurobi and PuLP

After generating churning probabilities, we then leveraged pandas, to create a structured DataFrame where each row represents a customer, and columns inside represents churning probability.We formulated our optimization model using Gurobi for model 1, and PuLP for model 2, both as optimization libraries available in Python that enables the definition of optimization problems with objectives and constraints. More on detailed model formulations will be explored in the section 4.

3.3., Visualization in MatPlotLib

Matplotlib, a comprehensive library for visualizations in Python, is utilized to graphically represent the promotion strategies derived from the optimization models. Through the creation of bar charts, pie charts, and additional plots, we provide visual insights into the promotion strategies through distribution of churn probabilities, and the proportion of customers receiving promotions.

4. Modeling

4.1. Model 1 – Basic Single Tier Promotion Strategy

Our first model, a single-tier basic optimization model ensures uniform promotion costs for all customers, regardless of individual characteristics such as balances. The model strikes a balance between proof-of-concept simplicity, while also ensuring effectiveness. This model was developed as a simple baseline for our more complex model.

As seen in Appendix A.1., this model determines if a customer will receive a promotion or not, using the decision variable x_i. The only other consideration is the cost of promotion per customer B, and the total budget for all promotions B_{max}. The objective function ensures that customers with the highest churn probabilities receive promotions, while the constraints ensure that the total promotions cost does not exceed the total budget.

4.2. Model 2 – Multiple Tiers Promotion Strategy with a Trade Off Parameter

This model acts as an extension from model 1. The mathematical formulation for this model can be found in Appendix B. Specifically, this model incorporates two key parameters: the budget (B) and the trade-off parameter (w), each playing a crucial role in optimizing promotional strategies for customer retention. The budget parameter (B) determines the total amount allocated for promotions, while the trade-off parameter (w) serves as a weighting factor that governs the relative importance of two key objectives in the optimization problem: minimizing churn probability among customers who do not receive promotions, and controlling the budget spent on promotions. This approach represents an improvement over the basic model by introducing a more nuanced strategy for allocating promotional resources.

Constraints ensure that the total promotion cost does not exceed the allocated budget and that each customer receives at most one promotion. Additionally, the model assumes that the promotional cost is fixed between \$100 and \$200 for each customer, represented by decision variables x_i^{100} and x_i^{200}. The promotion assignment assumes that customers with churn rates between the lower threshold (0.2) and upper threshold (0.4) will receive a \$100 promotion, while those above the upper threshold will receive a \$200 promotion.

5. Results and Sensitivity Analysis

5.1. Model 1 Results, Advantages, and Limitations

Model 1 was tested with a promotion cost of $500 per customer and a maximum budget of $1M. This yielded in 22.7% of the customers, with an average churn probability of 0.31, receiving a promotion (Appendix A, Figures A-1, A-2). This model has many advantages such as being easy to implement and interpret. Nevertheless, it has limitations, such as assuming a uniform promotion budget for all customers. This means that regardless of individual churn risk, customers receive the same promotion. Consequently, those with a lower churn risk may receive promotions that exceed their needs, leading to potential inefficiencies in resource allocation. Thus, limitations were addressed in Model 2.

5.2. Model 2 Results and Sensitivity Analysis

For model 2, the sensitivity analysis examines the impact of varying key input parameters, including the total budget B, the trade-off parameter w, on the promotion strategy's effectiveness in reducing churn rate. By systematically exploring different budget allocations and trade-off weightings, it is aimed at identifying optimal configurations that strike a balance between minimizing churn and maximizing budget utilization.

5.2.1. Change of Budget (B)

In the first subsection of the sensitivity analysis, the total budget B is varied to explore its impact on the promotion strategy. Consider budget values of $100K, $500K, $1M, $1.2M and $1.5M, Table 1 shows how different budget allocations affect the trade-off between minimizing churn rate and maximizing budget utilization. The visualized line plot for budget parameter sensitivity analysis can be seen in Figure B-1 for changing budget values.

As can be seen from Table B-1, with a budget of $1M, there is a notable decrease in the average churn probability of customers received promotions, indicating the most effective promotion strategies. Furthermore, this budget allows for a balanced allocation between the $100 and $200 promotions, ensuring a fair distribution of promotional benefits among customers. However, an increase in the budget to $1.5M results in a slight rise in the average churn probability but enables a larger proportion of customers to receive the higher-value $200 promotion.

5.2.2. Change of Trade-off Parameter (w)

This part of sensitivity analysis examines the impact of varying the trade-off parameter (w) on promotion effectiveness and customer churn rates with a total budget of $1M. By adjusting the weight assigned to the budget constraint versus the churn probability penalty, insights can be gained into the

optimal balance for maximizing customer retention within budget constraints. Detailed results can be found in Table B-2. The visualized line plot for trade-off parameter sensitivity analysis can be seen in Figure B-2.

According to Table B-2, the sensitivity analysis on the trade-off parameter (w) suggests that selecting a value of 0.5 would be advantageous for optimizing promotion effectiveness and resource allocation. It can also be seen that 0. 3 and 0.5 generate the same results. With value of 0.5, the average churn rate among customers with promotions is minimized, indicating enhanced retention efforts. Furthermore, the churn rate for customers offered the $100 promotion is among the lowest, indicating effective targeting of this promotion tier. Conversely, the churn rate for customers receiving the $200 promotion is the highest, suggesting that a larger proportion of high-value customers are being engaged with this offer. Therefore, a trade-off parameter of 0.5 strikes a balance between minimizing churn rates and strategically allocating promotions to maximize customer retention and value.

5.2.3. Model 2 Sensitivity Analysis Conclusion

After conducting sensitivity analyses on both the budget (B) and the trade-off parameter (w), it was determined that the optimal combination for maximizing customer retention and promotion effectiveness is a budget of $1M with a trade-off parameter of 0.5. This selection yielded the most favourable outcomes, as demonstrated by the resulting churn probabilities and distribution of promotion offers in Table B-3. As a result, in appendix B, Figure B-3, which is the pie chart representing the proportion of customer groups with different promotion strategies, it has 13.1% of total customers with no promotion and 12.4% with $200 promotion, and 74.5% with $100. Also shown in Figure B-4, the average churn probability of the customers who received a promotion is 0.23, which is higher than 0.02 for the average churn probability of customers with no promotion. Further can be seen from Figure B-5, with the blue dotted line representing the average churn probability for all customers which is 0.19, it clearly separates the 3 categories of promotion strategies. Customers with $200 have the highest churn probability of 34.59%, and 21.08% for customers with $100, 1.5% for customers with no promotion at all.

6. Conclusion

As a concluding remark, the report explores a proof-of-concept optimization model, with an implemented extension in attempting to minimize customer churn. The report explores how our extended model, implemented in PuLP minimizes churn probability for unpromoted customers, and minimizing budget for customers who receive promotion with budget (B) and trade-off parameter (w). After sensitivity analysis, a budget of $1M with a trade-off parameter of 0.5 is deemed optimal, yielding in 86.9% customers receiving promotion. Naturally, there are certain limitations of the

model, which includes pre-defined promotional values of $100 and $200, and budget of $1M. Each bank should adjust the budget to their deposit balances, while also adjusting the set promotional value of $100 and $200 as a percentage of their customers' balances, indicating interest expenses. That said, we believe our extended optimization model should act as a sufficient prescriptive analytics tool in helping banks decide who to give promotional offers towards.

References

ChatGPT. Chat.openai.com. https://chat.openai.com/c/925ea75e-1a54-4b5d-a575-51140b5351ae https://chat.openai.com/share/d1ec0fa7-05bd-4153-b0b3-1aa54de92154

Kishore, P. K. (n.d.). *Bank Customer Churn Data*. Www.kaggle.com. Retrieved February 29, 2024, from https://www.kaggle.com/datasets/pentakrishnakishore/bank-customer-churn-data/data?select=churn_prediction.csv

Marous J. *Recognizing "Silent Attrition" Is Key to Maintaining Loyalty in Banking*. (2023, April 23). Recognizing "Silent Attrition" Is Key to Maintaining Loyalty in Banking; The Financial Brand. https://thefinancialbrand.com/news/customer-experience-banking/silent-attrition-key-to- customer-loyalty-in-banking-161500/

Appendix A – Model 1

A.1. Model 1 Mathematical Formulation

Parameters

$$P_i \quad : Probability\ of\ customer\ i\ churning\ (from\ predictive\ analysis)$$

$$B \quad : Promotion\ budget\ per\ customer$$

$$B_{max} : Total\ promotion\ budget$$

Decision Variables

$$x_i = \begin{cases} 1 \; if \; customer \; i \; receives \; a \; promotion, \\ 0 \; otherwise \end{cases}$$

Objective Function

Minimizing the overall churn rate by maximizing the sum of probability of churn multiplied with x for each customer to ensure customers with the highest churn probability are given a promotion

$$max \sum_i P_i x_i \tag{1}$$

Constraints

$$\sum_i B_i x_i \leq B_{max} \tag{2}$$

$$x_i \in \{0, 1\} \tag{3}$$

A.2. Model 1 Results

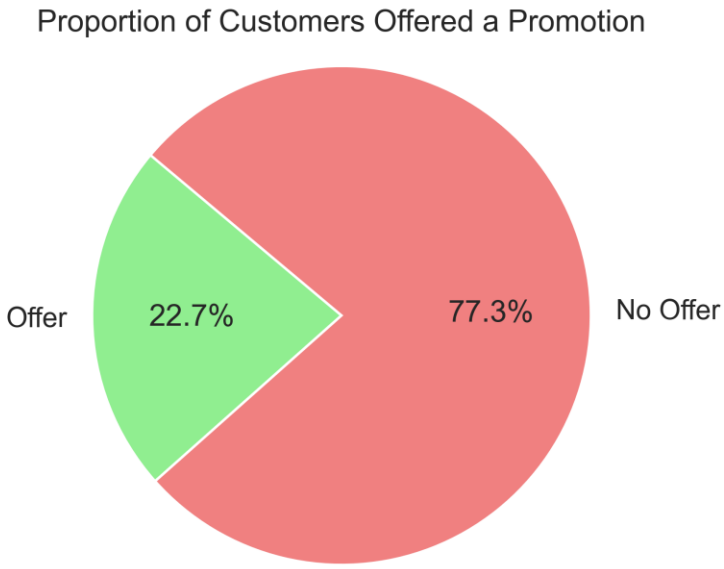


Figure A-1: Proportion of Customers Offered a Promotion (B = 500, B_{max}= 1 million)

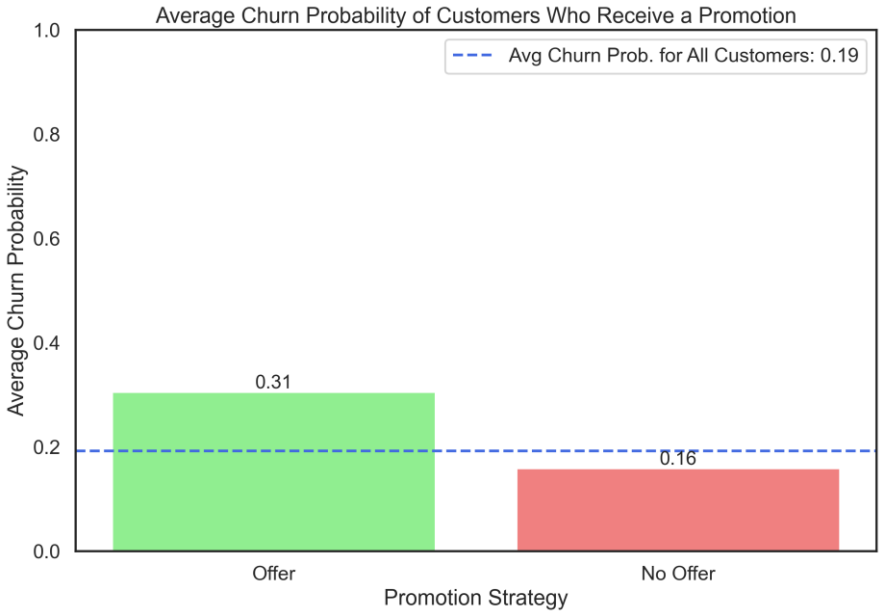


Figure A-2: Average Churn Probability of Each Customer Group (B = 500, B_{max}= 1 million)

Appendix B – Model 2

Parameters

B: $Total\ promotion\ budget$

P_i: $Probability\ of\ customer\ i\ churning\ (from\ predictive\ analysis)$

$\boldsymbol{C}$: $\boldsymbol{Cost\ of\ promotion} \in \{\mathbf{100}, \mathbf{200}\}$

$\boldsymbol{w}$: $\boldsymbol{Trade\ off\ parameter\ between\ reducing\ churn\ and\ optimizing\ the\ budget}$

Decision Variables

$x_i^{100} = 1$: $customer\ i\ receives\ a\ promotion\ 100$

$x_i^{200} = 1$: $customer\ i\ receives\ a\ promotion\ 200$

Objective Function

Minimizing Churn Probability for Unpromoted Customers & Minimize Total Budget Spent for Customers Receive Promotions

$$min\ \ w * \sum_i P_i * (1 - (x_i^{100} + x_i^{200}) + (1 - w) * \frac{\sum_i (x_i^{100} * C_{100} + x_i^{200} * C_{200})}{B} \quad (1)$$

Constraints

$$\sum_i (x_i^{100} * C_{100} + x_i^{200} * C_{200}) \leq B\ \ Total\ Budget \quad (2)$$

$$x_i^{100} + x_i^{200} \leq 1 \quad No\ more\ than\ 1\ promotion\ per\ customer \quad (3)$$

$$0 \leq w \leq 1,\ x_i^{100} \in \{0, 1\},\ x_i^{200} \in \{0, 1\} \quad (4)$$

Table B-1: Sensitivity Analysis for Budget (B) with $w = 0.5$

	Average churn probability of customers	Proportion of customers

Budget (B)	with promotion	with $100 promotion	with $200 promotion	with $100 promotion	with $200 promotion	with no promotion
$100,000	0.26	0.24	0.57	0.10	0.007	0.89
$500,000	0.27	0.25	0.54	0.50	0.03	0.50
$1,000,000	0.23	0.21	0.35	0.85	0.14	0.15
$1,200,000	0.23	0.21	0.30	0.89	0.23	0.10
$1,500,000	0.22	0.20	0.27	0.97	0.36	0.03

Table B-2: Sensitivity Analysis for Trade-off Parameter (w) with $1M Budget

Trade-off Parameter (w)	Average churn probability of customers			Proportion of customers		
	with promotion	with $100 promotion	with $200 promotion	with $100 promotion	with $200 promotion	with no promotion
0	0.27	0.25	0.59	0.34	0.024	0.63
0.3	0.23	0.21	0.35	0.85	0.14	0.15
0.5	0.23	0.21	0.35	0.85	0.14	0.15
1	0.23	0.21	0.35	0.85	0.14	0.15

Table B-3: Result for Optimal Selection of Parameters

Parameters	Average Churn Probability of Customers			Proportion of Customers		
B = $1M w = 0.5	With promotion	With $100 promotion	With $200 promotion	With $100 promotion	With $200 promotion	With no promotion
	0.23	0.21	0.35	0.85	0.14	0.15

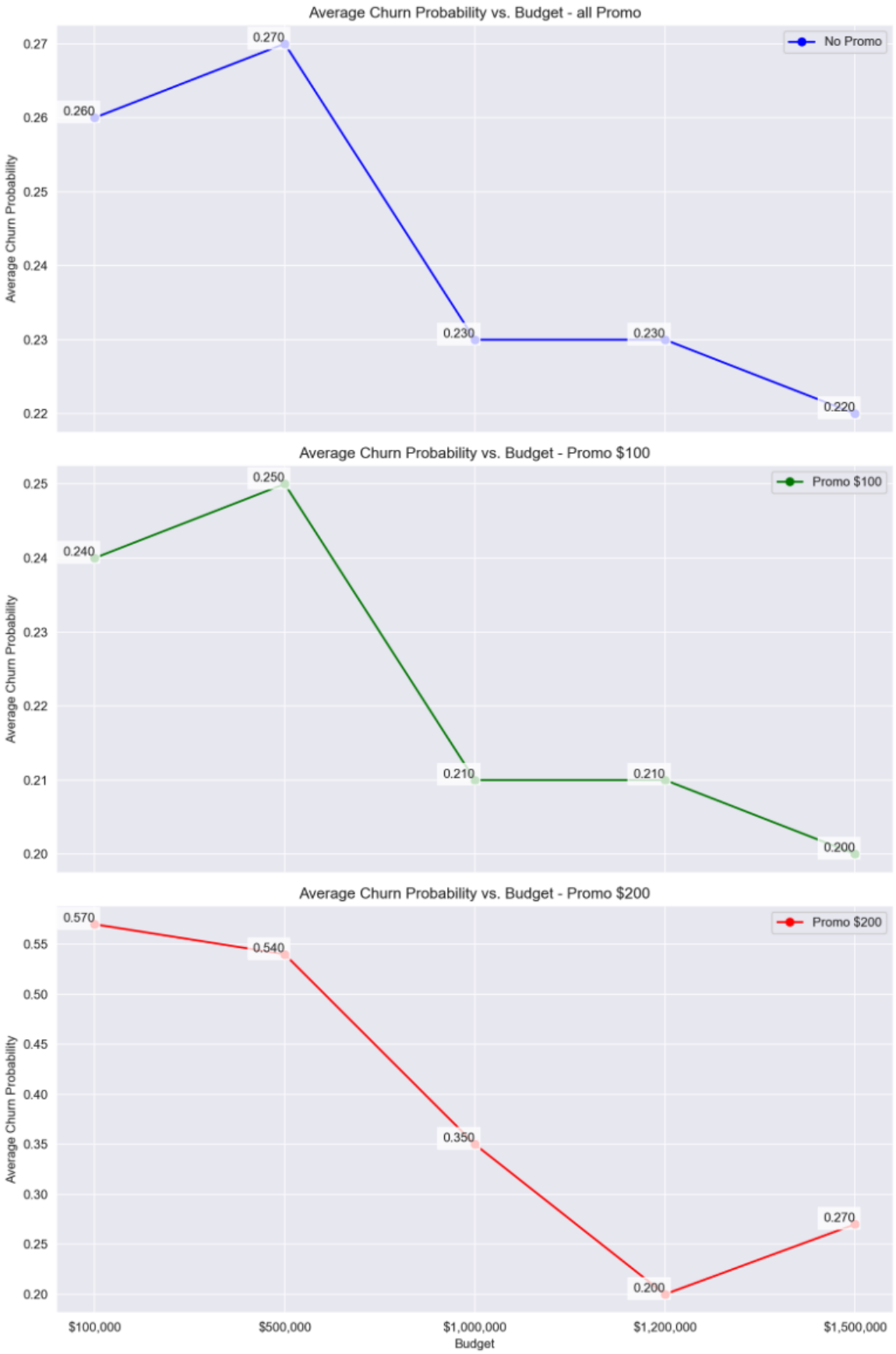


Figure B-1: Visualized Results for Budget Parameter Sensitivity Analysis

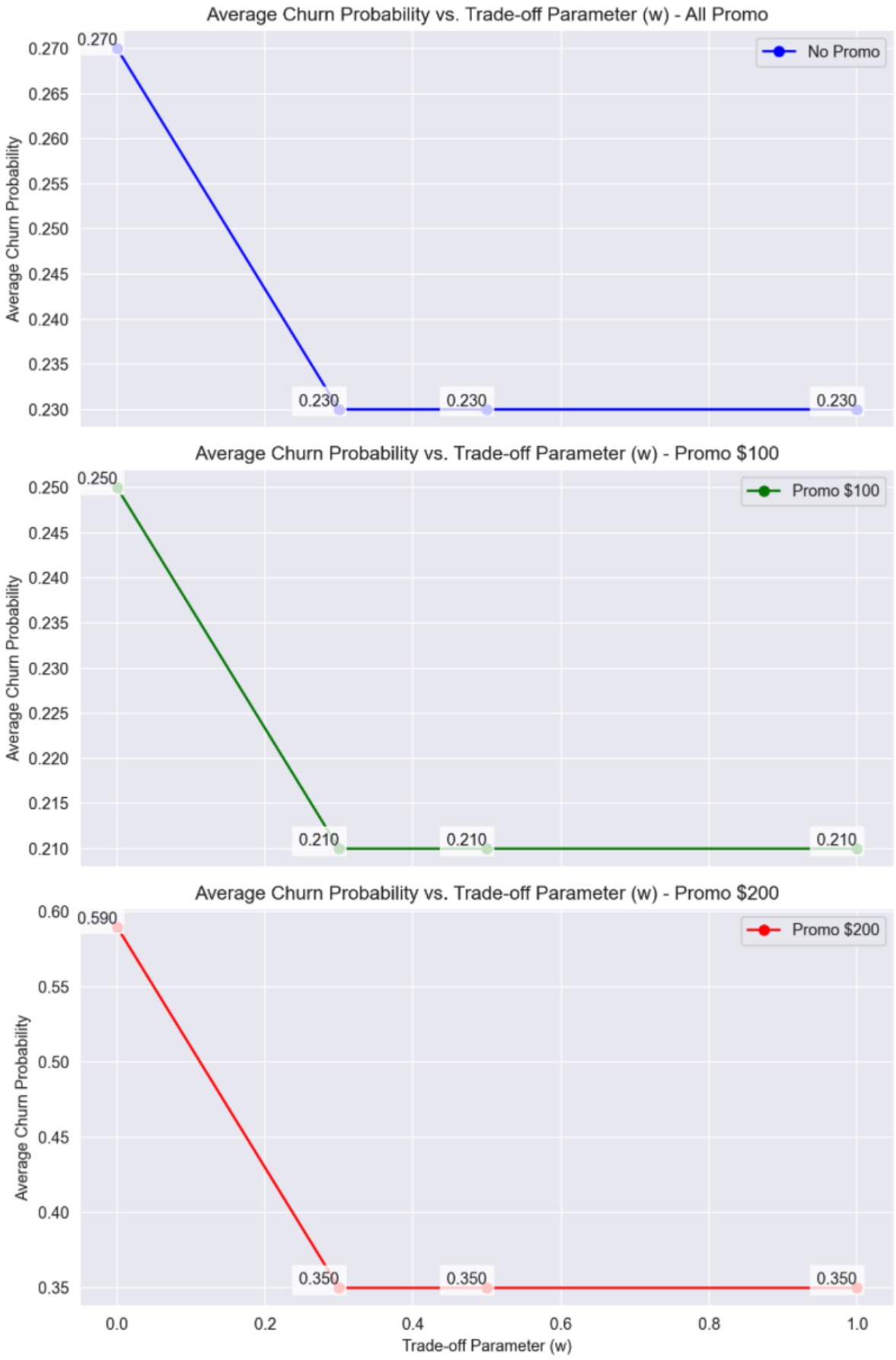


Figure B-2: Visualized Results for Trade-off Parameter Sensitivity Analysis

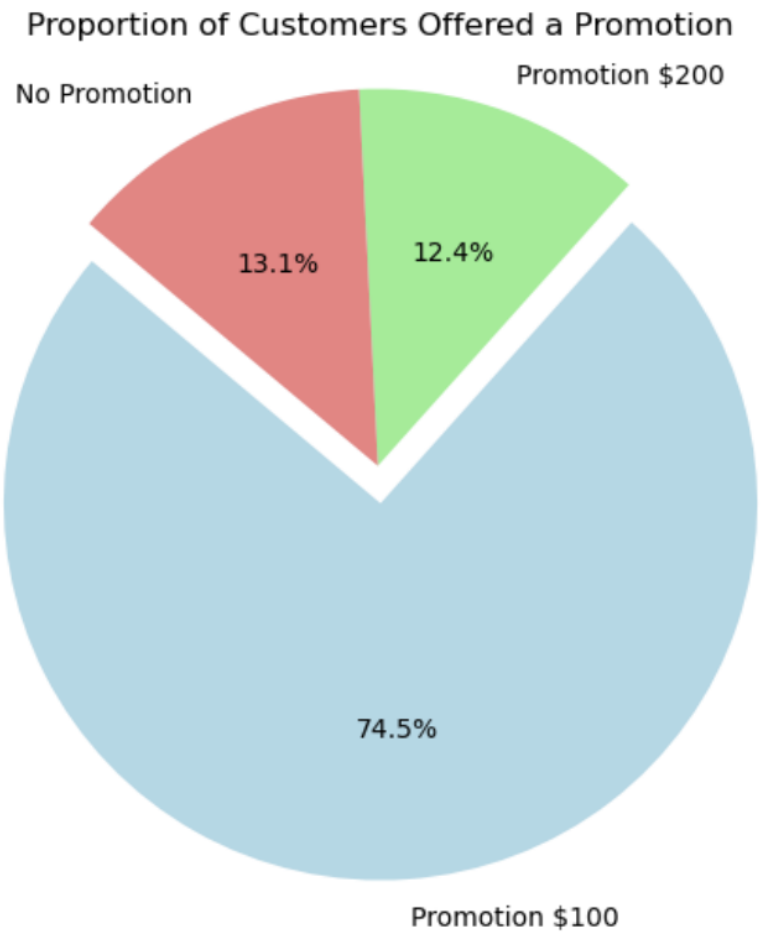


Figure B-3: Proportion of Customers with Different Strategies

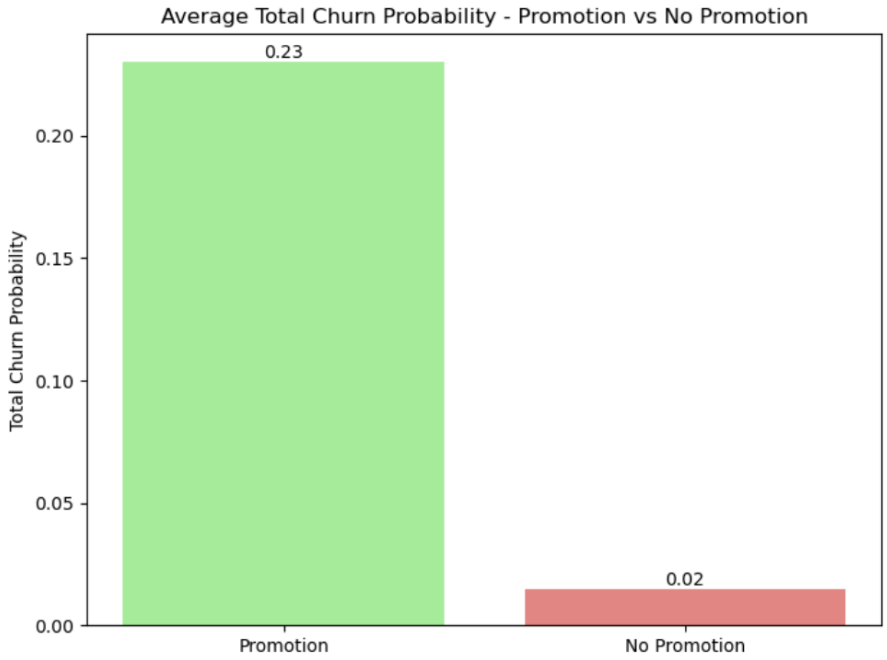


Figure B-4: Average Total Churn Probability - Promotion vs No Promotion

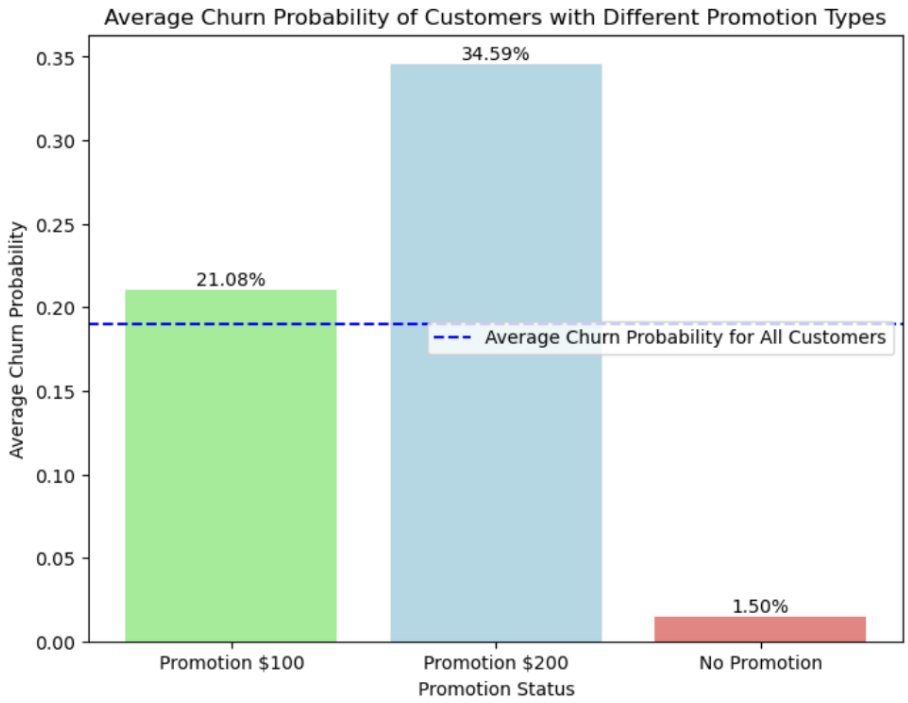


Figure B-5 Average Churn Probability of Customers with Different Promotion Types

Appendix C – Code

C.1. Model 1 Code

```
import pandas as pd
df = pd.read_csv('predictions.csv')
churn_probabilities = df['Churn_Probability'].values
print(churn_probabilities)
```

```
# uncomment this and replace with your license if you need to import it this way
# import os
# # Set the GRB_LICENSE_FILE environment variable
# os.environ['GRB_LICENSE_FILE'] = '/Users/nayeema/GUROBI/gurobi.lic'
```

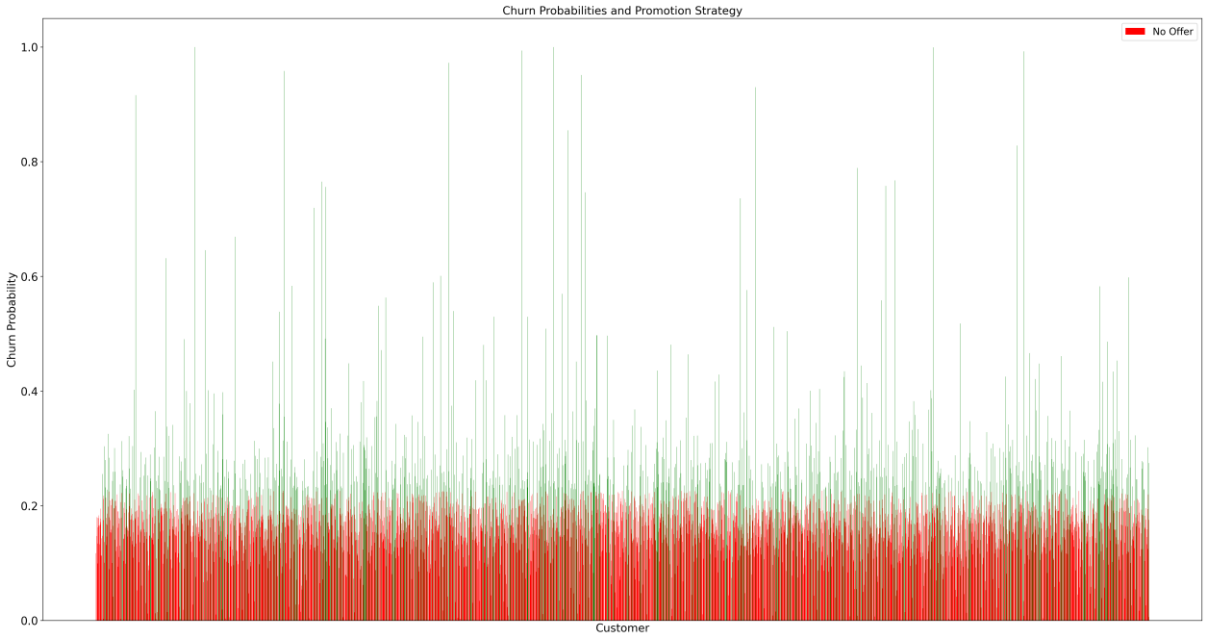
```
import matplotlib.pyplot as plt

def plot_solution(x,num_customers,churn_probabilities):
    # Determine promotion strategy for each customer

    promotion_strategy = ["Offer" if x[i].x > 0.5 else "No Offer" for i in range(num_customers)]

    # Plot churn probabilities and promotion strategy
    plt.figure(figsize=(30, 16), dpi=200)
    plt.bar(range(num_customers), churn_probabilities, color=['green' if s == 'Offer' else 'red' for s in promotion_strategy])
    plt.xlabel('Customer', fontsize=16)
    plt.ylabel('Churn Probability', fontsize=16)
    plt.title('Churn Probabilities and Promotion Strategy', fontsize=16)
    plt.xticks([])
    plt.yticks(fontsize=16)
    plt.legend(['No Offer', 'Offer'], loc='upper right', fontsize=14)
    plt.show()
```

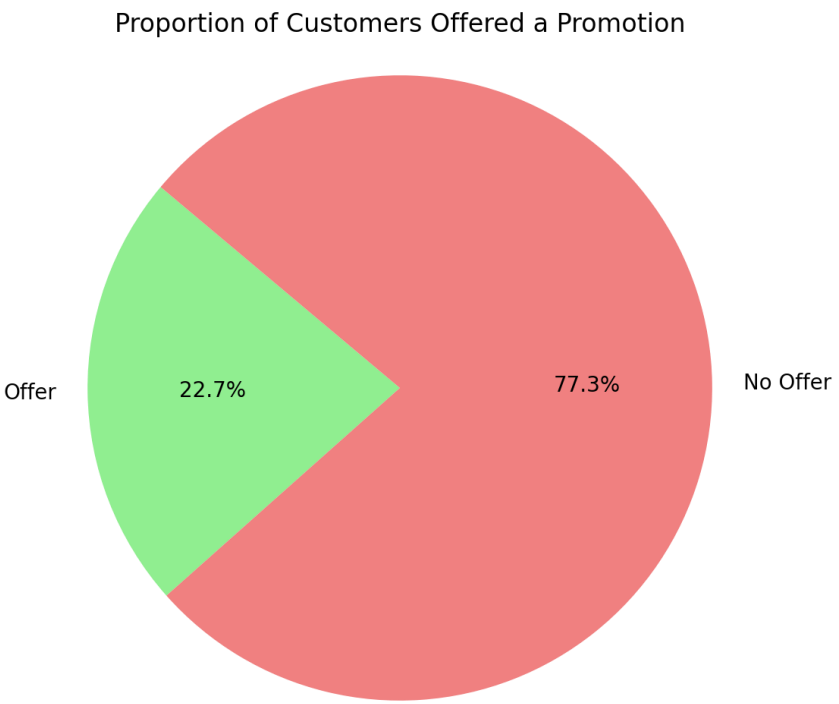
```
    # Count the number of customers who receive a promotion
    num_customers_offer = sum(1 for strategy in promotion_strategy if strategy == "Offer")
    num_customers_no_offer = len(promotion_strategy) - num_customers_offer

    # Calculate the average churn probability of customers who receive a promotion
    avg_churn_prob_offer = sum(churn_probabilities[i] for i in range(len(x)) if promotion_strategy[i] == "Offer") /
num_customers_offer
    # Calculate the average churn probability of customers who do not receive a promotion
    avg_churn_prob_no_offer = sum(churn_probabilities[i] for i in range(len(x)) if promotion_strategy[i] == "No Offer") /
num_customers_no_offer


    # Pie chart: Proportion of customers who receive a promotion vs. those who do not
    labels = ['Offer', 'No Offer']
    sizes = [num_customers_offer, num_customers_no_offer]
    colors = ['lightgreen', 'lightcoral']
    plt.figure(figsize=(10, 6), dpi=200)
    plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', shadow=False, startangle=140)
    plt.title('Proportion of Customers Offered a Promotion')
    plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
    plt.show()

    # Bar chart: Average churn probability of customers who receive a promotion
    plt.figure(figsize=(10, 6), dpi=200)
    barplot = plt.bar(['Offer', 'No Offer'], [avg_churn_prob_offer, avg_churn_prob_no_offer], color=['lightgreen', 'lightcoral'])
    plt.xlabel('Promotion Strategy')
    plt.ylabel('Average Churn Probability')
    plt.title('Average Churn Probability of Customers Who Receive a Promotion')
    plt.ylim(0, 1)
    for bar in barplot:
        yval = bar.get_height()
        plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), ha='center', va='bottom', fontsize=12)  # Add the
numbers on top of the bars
    plt.show()
```

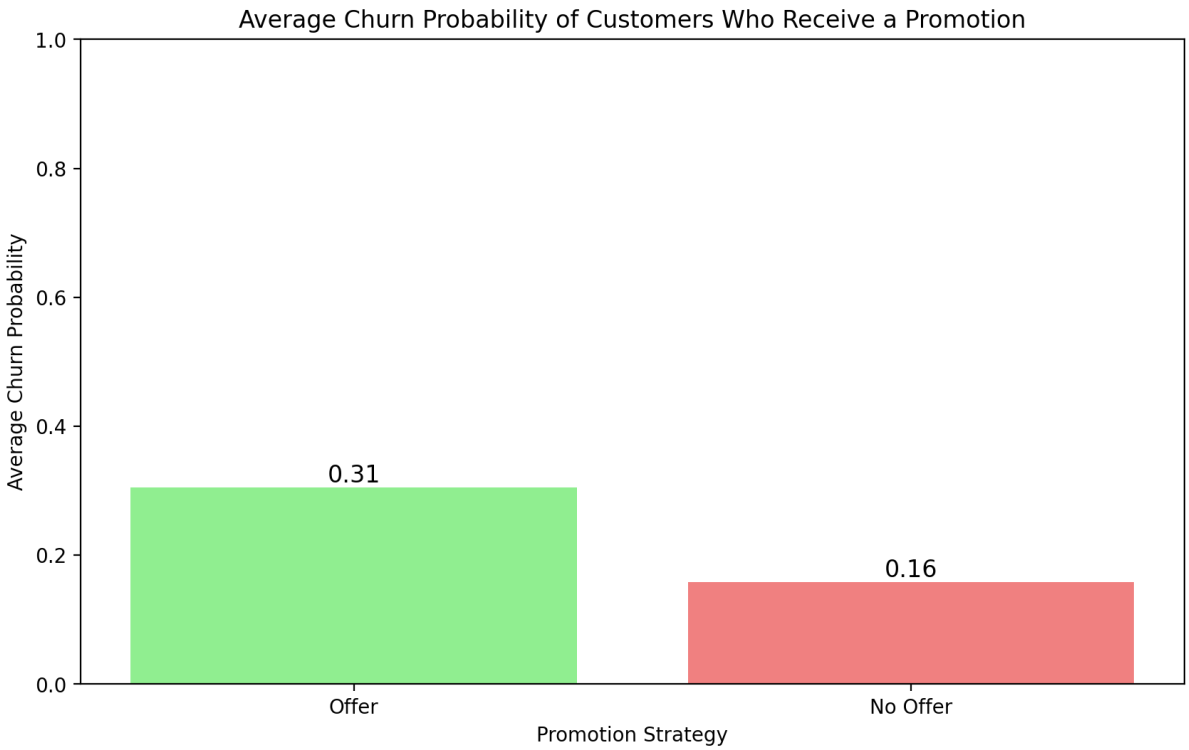
```
import gurobipy as gp
from gurobipy import GRB


budget_per_customer = 500
max_budget = 1000000


model = gp.Model('PromotionOptimization')

num_customers = len(churn_probabilities)
x = model.addVars(num_customers, vtype=GRB.BINARY, name="promotion")
model.setObjective(sum(churn_probabilities[i] * x[i] for i in range(num_customers)), GRB.MAXIMIZE)
model.addConstr(sum(budget_per_customer * x[i] for i in range(num_customers)) <= max_budget, "Budget")

model.optimize()

# if model.status == GRB.OPTIMAL:
#     print("Optimal promotion strategy:")
```

```
#     for i in range(num_customers):
#         if x[i].x > 0.5:
#             print(f"Customer {i}: Offer promotion")

plot_solution(x,num_customers,churn_probabilities)
```

Set parameter Username Academic license - for non-commercial use only - expires 2025-03-22 Gurobi Optimizer version 10.0.2 build v10.0.2rc0 (mac64[rosetta2]) CPU model: Apple M1 Thread count: 8 physical cores, 8 logical processors, using up to 8 threads Optimize a model with 1 rows, 8820 columns and 8820 nonzeros Model fingerprint: 0xdd55c97c Variable types: 0 continuous, 8820 integer (8820 binary) Coefficient statistics: Matrix range [5e+02, 5e+02] Objective range [2e-16, 1e+00] Bounds range [1e+00, 1e+00] RHS range [1e+06, 1e+06] Found heuristic solution: objective 385.7386980 Presolve removed 0 rows and 8 columns Presolve time: 0.01s Presolved: 1 rows, 8812 columns, 8812 nonzeros Variable types: 0 continuous, 8812 integer (8812 binary) Found heuristic solution: objective 403.7774668 Root relaxation: objective 6.108111e+02, 1 iterations, 0.00 seconds (0.00 work units) Nodes | Current Node | Objective Bounds | Work Expl Unexpl | Obj Depth IntInf | Incumbent BestBd Gap | It/Node Time * 0 0 0 610.8110670 610.81107 0.00% - 0s Explored 1 nodes (1 simplex iterations) in 0.04 seconds (0.02 work units) Thread count was 8 (of 8 available processors) Solution count 3: 610.811 403.777 385.739 Optimal solution found (tolerance 1.00e-04) Best objective 6.108110670082e+02, best bound 6.108110670082e+02, gap 0.0000%







C.2. Model 2 Code

```
from pulp import LpVariable, LpProblem, LpMinimize, lpSum, value

# Parameters
B = 1000000  # Total promotion budget
C = {100: 100, 200: 200}  # Cost of promotions
w = 0.5  # Weight assigned to the budget constraint
lower_threshold = 0.2  # Lower churn rate threshold
upper_threshold = 0.4  # Upper churn rate threshold
```

```
# Decision Variables
customers = range(len(churn_probabilities))
x_100 = LpVariable.dicts("x_100", customers, cat="Binary")  # Customer receives promotion $100
x_200 = LpVariable.dicts("x_200", customers, cat="Binary")  # Customer receives promotion $200

# Objective Function
model = LpProblem("CustomerPromotionOptimization", LpMinimize)
model += w * lpSum(churn_probabilities[i] * (1 - (x_100[i] + x_200[i])) for i in customers) \
        + (1 - w) * (lpSum(x_100[i] * C[100] + x_200[i] * C[200] for i in customers) / B)

# Constraints
model += lpSum(x_100[i] * C[100] + x_200[i] * C[200] for i in customers) <= B, "TotalPromotionBudget"
for i in customers:
    if churn_probabilities[i] > upper_threshold:
        model += x_200[i] == 1, f"Promotion200_{i}"
    elif lower_threshold <= churn_probabilities[i] <= upper_threshold:
        model += x_100[i] == 1, f"Promotion100_{i}"
    else:
        model += x_200[i] == 0, f"NoPromotion_{i}"

# Solve the optimization problem
model.solve()

# Output results
print("Objective Value:", value(model.objective))

# Print decision variables x_100 and x_200
print("Values of decision variables x_100 and x_200:")
for i in customers:
    print("Customer", i, "receives promotion $100:", value(x_100[i]))
    print("Customer", i, "receives promotion $200:", value(x_200[i]))
```

```
Welcome to the CBC MILP Solver Version: 2.10.3 Build Date: Dec 15 2019 command line - /Users/yalichen/anaconda3/lib/python3.11/site-packages/pulp/solverdir/cbc/osx/64/cbc /var/folders/1w/6whx73nx5r5cndzqxqq4d31w0000gn/T/74230a0b7d414dbe9f15a8952fdd7282-pulp.mps -timeMode elapsed -branch -printingOptions all -solution /var/folders/1w/6whx73nx5r5cndzqxqq4d31w0000gn/T/74230a0b7d414dbe9f15a8952fdd7282-pulp.sol (default strategy 1) At line 2 NAME MODEL At line 3 ROWS At line 8826 COLUMNS At line 88207 RHS At line 97029 BOUNDS At line 114670 ENDATA Problem MODEL has 8821 rows, 17640 columns and 26460 elements Coin0008I MODEL read with 0 errors Option for timeMode changed from cpu to elapsed Continuous objective value is -1409.15 - 0.01 seconds Cgl0002I 5594 variables fixed Cgl0004I processed model has 1 rows, 8797 columns (8797 integer (8797 of which binary)) and 8797 elements Cbc0038I Initial state - 0 integers unsatisfied sum - 0 Cbc0038I Solution found of -1409.15 Cbc0038I Before mini branch and bound, 8797 integers at bound fixed and 0 continuous Cbc0038I Mini branch and bound did not improve solution (0.62 seconds) Cbc0038I After 0.62 seconds - Feasibility pump exiting with objective of -1409.15 - took 0.01 seconds Cbc0012I Integer solution of -1409.1488 found by feasibility pump after 0 iterations and 0 nodes (0.62 seconds) Cbc0001I Search completed - best objective -1409.148800103356, took 0 iterations and 0 nodes (0.62 seconds) Cbc0035I Maximum depth 0, 0 variables fixed on reduced cost Cuts at root node changed objective from -1409.15 to -1409.15 Probing was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds) Gomory was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds) Knapsack was tried 0 times and created 0 cuts of which 0 were active
```

after adding rounds of cuts (0.000 seconds) Clique was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds) MixedIntegerRounding2 was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds) FlowCover was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds) TwoMirCuts was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds) ZeroHalf was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds) Result - Optimal solution found Objective value: -1409.14880010 Enumerated nodes: 0 Total iterations: 0 Time (CPU seconds): 0.62 Time (Wallclock seconds): 0.63 Option for printingOptions changed from normal to all Total time (CPU seconds): 0.66 (Wallclock seconds): 0.67 Objective Value: -223.19387422659418

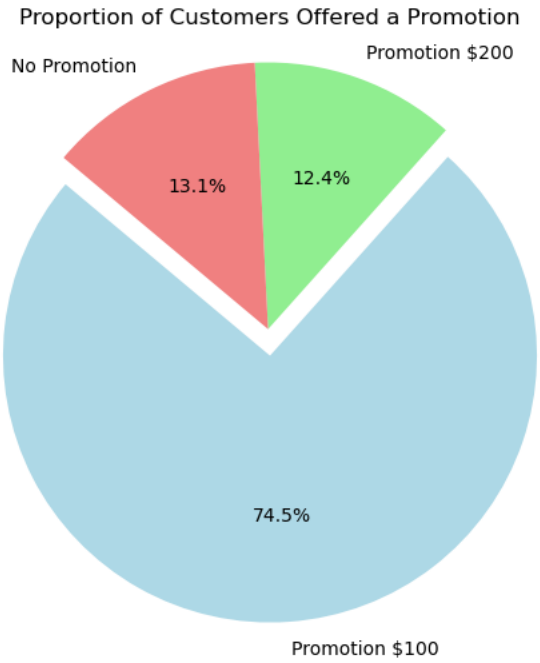
```
# Calculate total churn probability for customers who received promotions ($100 and $200)
total_churn_prob_promotions = sum(churn_probabilities[i] for i in customers if value(x_100[i]) + value(x_200[i]) > 0)
# Count the number of customers who received each promotion type
num_customers_promotion_100 = sum(1 for i in customers if value(x_100[i]))
num_customers_promotion_200 = sum(1 for i in customers if value(x_200[i]))
num_customers_no_promotion = sum(1 for i in customers if not value(x_100[i]) and not value(x_200[i]))
# Calculate total churn probability for customers with $200 promotion
total_churn_prob_200_promotion = sum(churn_probabilities[i] for i in customers if value(x_200[i]))
# Calculate total churn probability for customers with $100 promotion
total_churn_prob_100_promotion = sum(churn_probabilities[i] for i in customers if value(x_100[i]))
#
sum12 = total_churn_prob_200_promotion+total_churn_prob_100_promotion

avg_churn_prob_all_promotions = sum12 / (num_customers_promotion_100+num_customers_promotion_200)

# Calculate -
avg_churn_prob_100 =  total_churn_prob_100_promotion / num_customers_promotion_100

# Calculate the average churn probability for $200 promotion
avg_churn_prob_200 = total_churn_prob_200_promotion / num_customers_promotion_200

# Calculate the average churn probability for no promotion
avg_churn_prob_no_promotion = sum(churn_probabilities[i] for i in customers if not value(x_100[i]) and not
value(x_200[i])) / (len(customers) - num_customers_no_promotion)

# Calculate the proportions
total_customers = len(customers)
prop_promotion_100 = num_customers_promotion_100 / total_customers
prop_promotion_200 = num_customers_promotion_200 / total_customers
prop_no_promotion = num_customers_no_promotion / total_customers

print(avg_churn_prob_all_promotions)
print(avg_churn_prob_100)
print(avg_churn_prob_200)
print(prop_promotion_100)
print(prop_promotion_200)
print(prop_no_promotion)
```

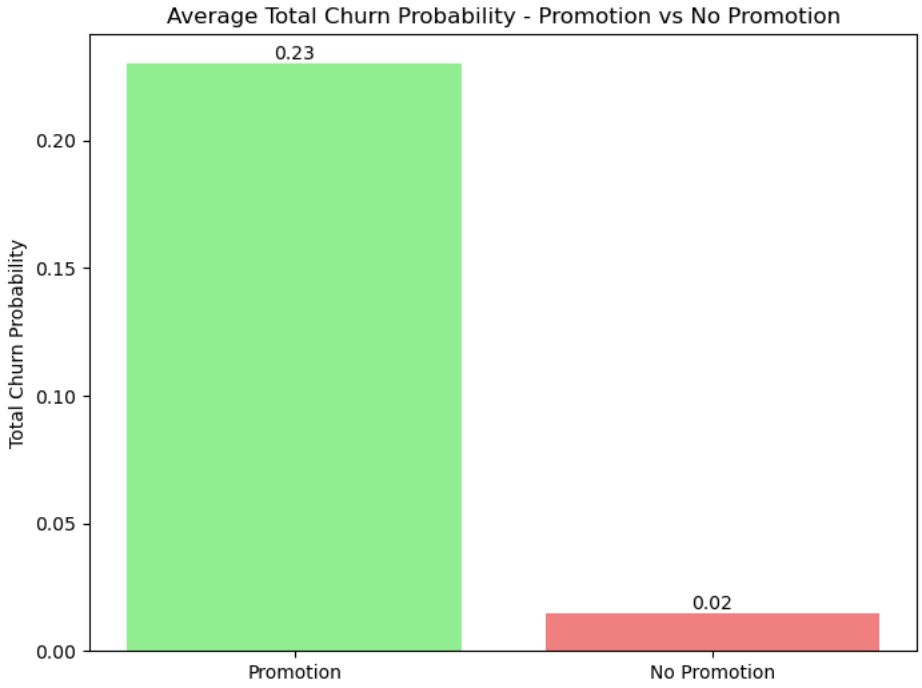
0.23008779406489904 0.21081259861096738 0.3458624268074092 0.8505668934240362 0.14160997732426303 0.14943310657596373

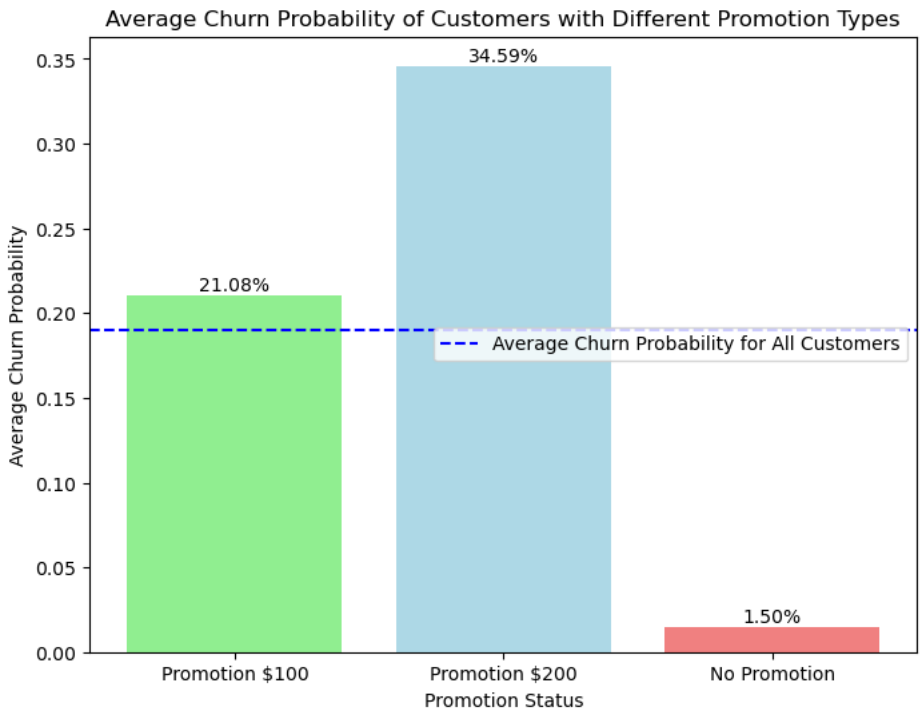
```
import matplotlib.pyplot as plt
import numpy as np
```

```
# Create data for Pie chart
labels = ['Promotion $100', 'Promotion $200', 'No Promotion']
sizes = [prop_promotion_100, prop_promotion_200, prop_no_promotion]
colors = ['lightblue', 'lightgreen', 'lightcoral']
explode = (0.1, 0, 0)  # explode 1st slice

# Create Pie chart
plt.figure(figsize=(8, 6))
plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)
plt.title('Proportion of Customers Offered a Promotion')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()



# Create data for stacked bar chart
categories = ['Promotion', 'No Promotion']
churn_probabilities_stacked = [avg_churn_prob_all_promotions, avg_churn_prob_no_promotion]

# Create Stacked bar chart
plt.figure(figsize=(8, 6))
bars = plt.bar(categories, churn_probabilities_stacked, color=['lightgreen', 'lightcoral'])
plt.title('Average Total Churn Probability - Promotion vs No Promotion')
plt.ylabel('Total Churn Probability')

# Add data labels on top of the bars
for bar, churn_prob in zip(bars, churn_probabilities_stacked):
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(), f'{churn_prob:.2f}',
             va='bottom', ha='center')

plt.show()



# Calculate average churn probability for each group
avg_churn_probs = [avg_churn_prob_100, avg_churn_prob_200, avg_churn_prob_no_promotion]

# Create data for Bar chart
labels = ['Promotion $100', 'Promotion $200', 'No Promotion']
colors = ['lightgreen', 'lightblue', 'lightcoral']

# Create Bar chart
plt.figure(figsize=(8, 6))
bars = plt.bar(labels, avg_churn_probs, color=colors)
plt.title('Average Churn Probability of Customers with Different Promotion Types')
plt.xlabel('Promotion Status')
plt.ylabel('Average Churn Probability')

# Add data labels on top of the bars
for bar, avg_churn_prob in zip(bars, avg_churn_probs):
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(), f'{avg_churn_prob:.2%}',
             va='bottom', ha='center')
```

```
# Add a line for average churn probability for all customers
plt.axhline(y=0.19, color='blue', linestyle='--', label='Average Churn Probability for All Customers')
plt.legend(loc='center right')

plt.show()
```







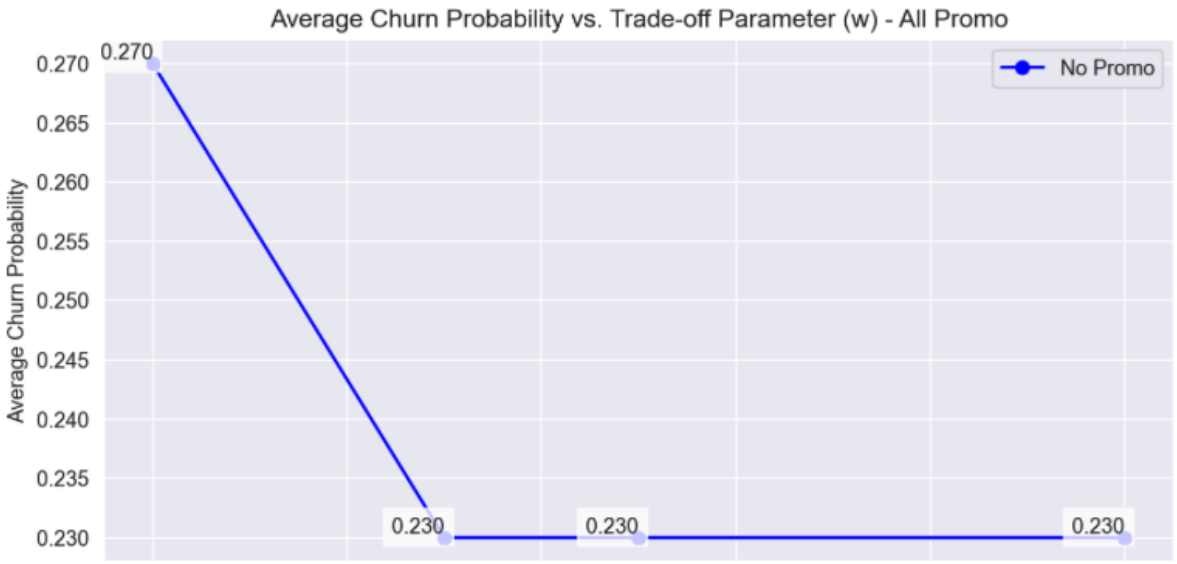

```
# Calculate the total budget used
total_budget_used = sum(value(x_100[i]) * C[100] + value(x_200[i]) * C[200] for i in customers)

# Print the total budget used
print("Total Budget Used:", total_budget_used)
```

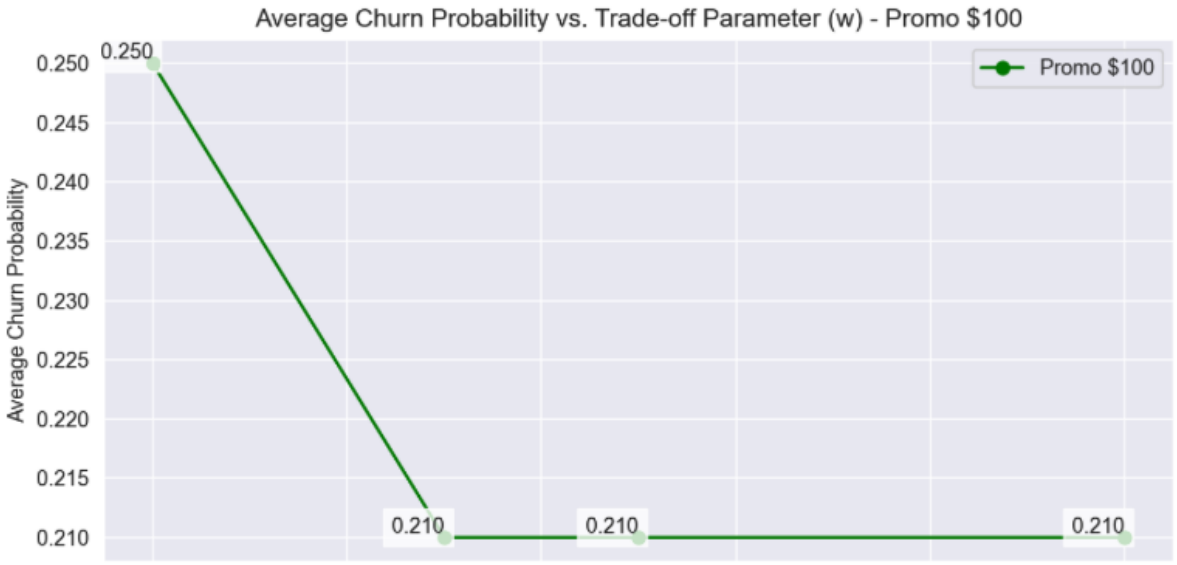
Total Budget Used: 1000000.0

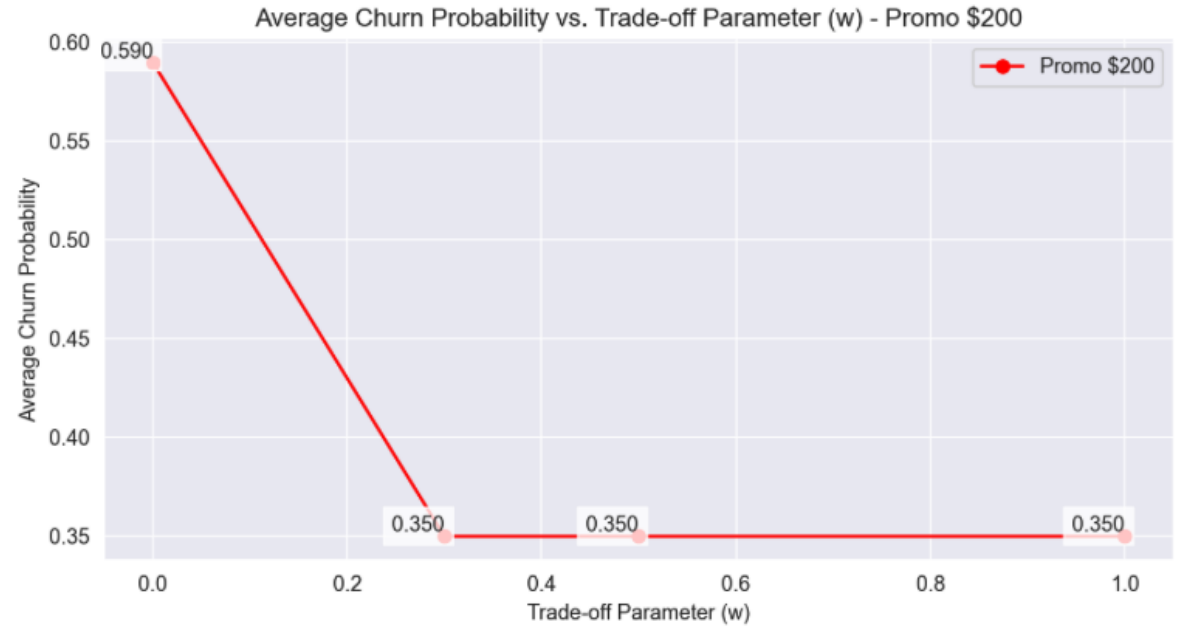
C.3. Sensitivity Analysis Results Plotting

```
import matplotlib.pyplot as plt
import seaborn as sns

# Set seaborn style
sns.set_style("darkgrid")

# Data
w_values = [0, 0.3, 0.5, 1]
avg_churn_no_promo = [0.27, 0.23, 0.23, 0.23]
avg_churn_100_promo = [0.25, 0.21, 0.21, 0.21]
avg_churn_200_promo = [0.59, 0.35, 0.35, 0.35]
# Create subplots
fig, axs = plt.subplots(3, 1, figsize=(8, 12), sharex=True, dpi=170)

# Plotting
axs[0].plot(w_values, avg_churn_no_promo, marker='o', color='blue', label='No Promo')
for x, y in zip(w_values, avg_churn_no_promo):
    axs[0].text(x, y, f'{y:.3f}', ha='right', va='bottom', bbox=dict(facecolor='white', alpha=0.8))
axs[0].set_title('Average Churn Probability vs. Trade-off Parameter (w) - No Promo')
axs[0].set_ylabel('Average Churn Probability')
```

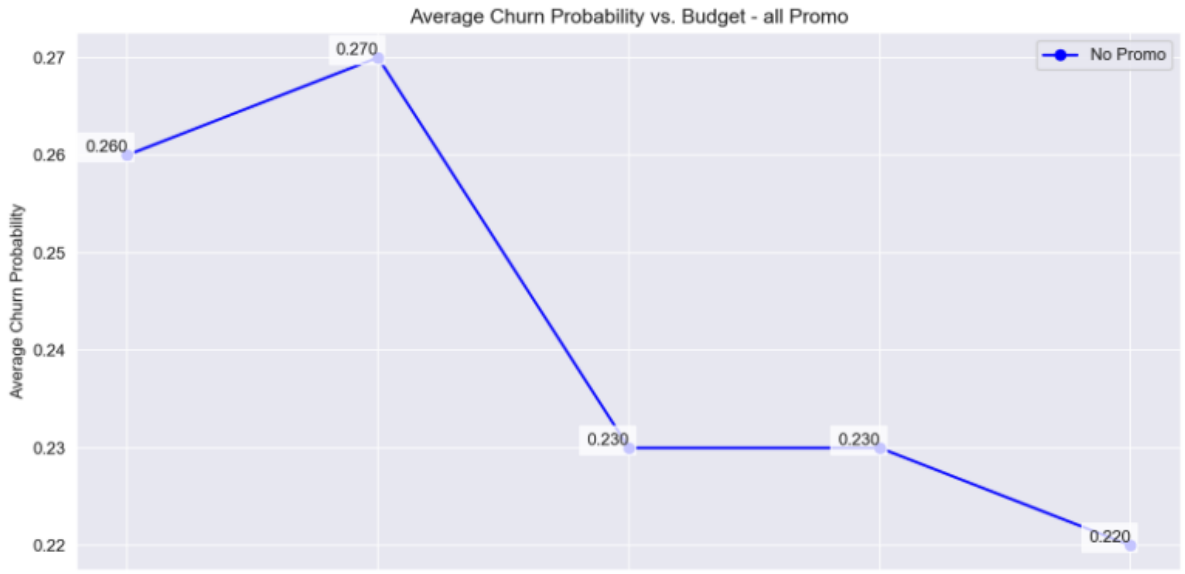
```
axs[0].legend()
axs[0].grid(True)

axs[1].plot(w_values, avg_churn_100_promo, marker='o', color='green', label='Promo $100')
for x, y in zip(w_values, avg_churn_100_promo):
    axs[1].text(x, y, f'{y:.3f}', ha='right', va='bottom', bbox=dict(facecolor='white', alpha=0.8))
axs[1].set_title('Average Churn Probability vs. Trade-off Parameter (w) - Promo $100')
axs[1].set_ylabel('Average Churn Probability')
axs[1].legend()
axs[1].grid(True)

axs[2].plot(w_values, avg_churn_200_promo, marker='o', color='red', label='Promo $200')
for x, y in zip(w_values, avg_churn_200_promo):
    axs[2].text(x, y, f'{y:.3f}', ha='right', va='bottom', bbox=dict(facecolor='white', alpha=0.8))
axs[2].set_title('Average Churn Probability vs. Trade-off Parameter (w) - Promo $200')
axs[2].set_xlabel('Trade-off Parameter (w)')
axs[2].set_ylabel('Average Churn Probability')
axs[2].legend()
axs[2].grid(True)

plt.tight_layout()
plt.show()
```

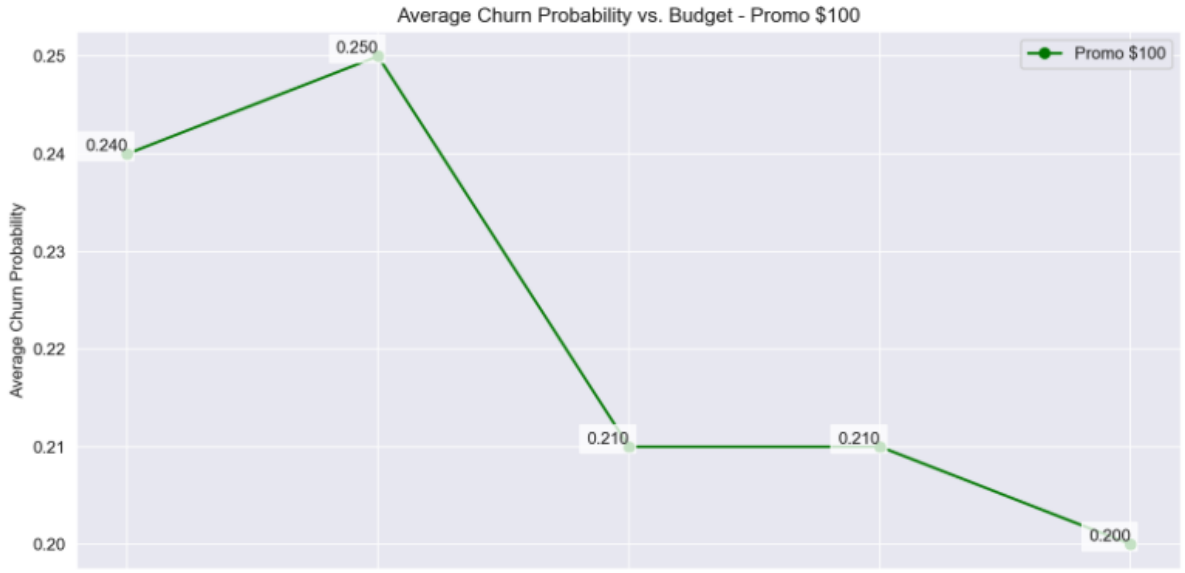

Average Churn Probability vs. Trade-off Parameter (w) - All Promo
No Promo
Average Churn Probability
0.270
0.265
0.260
0.255
0.250
0.245
0.240
0.235
0.230
0.270
0.230
0.230
0.230

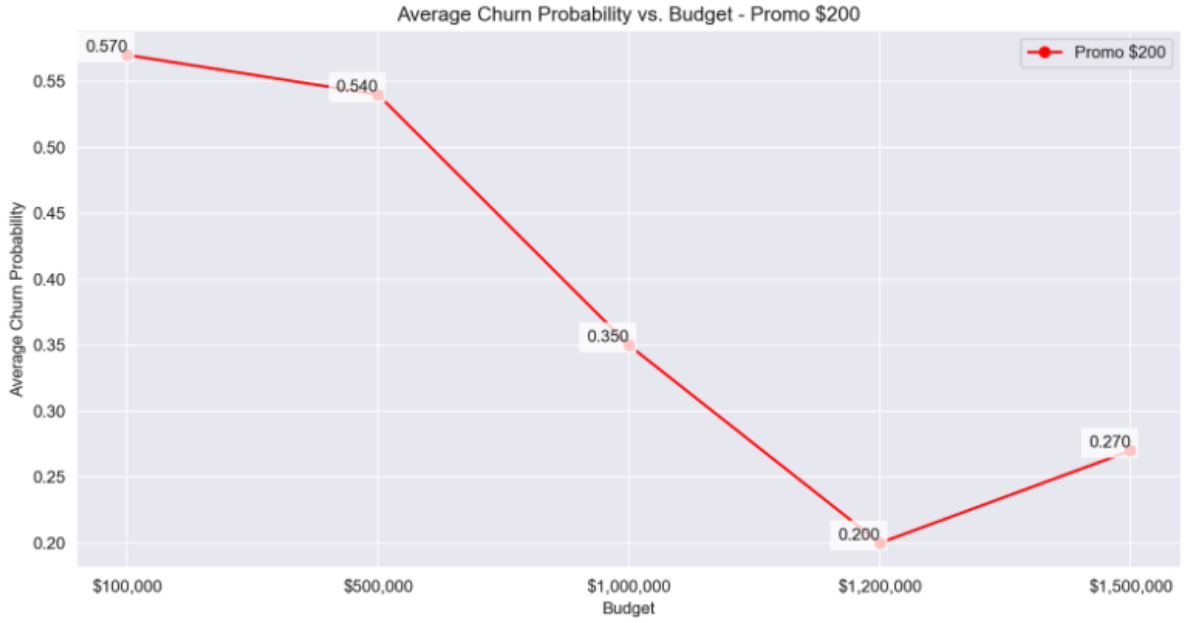

Average Churn Probability vs. Trade-off Parameter (w) - Promo $100
Promo $100
Average Churn Probability
0.250
0.245
0.240
0.235
0.230
0.225
0.220
0.215
0.210
0.250
0.210
0.210
0.210


Average Churn Probability vs. Trade-off Parameter (w) - Promo $200
Promo $200
Average Churn Probability
0.60
0.55
0.50
0.45
0.40
0.35
0.590
0.350
0.350
0.350
0.0
0.2
0.4
0.6
0.8
1.0
Trade-off Parameter (w)

```
import matplotlib.pyplot as plt
import seaborn as sns

# Set seaborn style
sns.set_style("darkgrid")

# Data
budgets = ['$100,000', '$500,000', '$1,000,000', '$1,200,000', '$1,500,000']
avg_churn_no_promo_b = [0.26, 0.27, 0.23, 0.23, 0.22]
avg_churn_100_promo_b = [0.24, 0.25, 0.21, 0.21, 0.20]
avg_churn_200_promo_b = [0.57, 0.54, 0.35, 0.20, 0.27]
# Create subplots
fig, axs = plt.subplots(3, 1, figsize=(10, 15), sharex=True, dpi=160)

# Plotting
axs[0].plot(budgets, avg_churn_no_promo_b, marker='o', color='blue', label='No Promo')
for x, y in zip(budgets, avg_churn_no_promo_b):
    axs[0].text(x, y, f'{y:.3f}', ha='right', va='bottom', bbox=dict(facecolor='white', alpha=0.8))
axs[0].set_title('Average Churn Probability vs. Budget - No Promo')
axs[0].set_ylabel('Average Churn Probability')
axs[0].legend()
axs[0].grid(True)

axs[1].plot(budgets, avg_churn_100_promo_b, marker='o', color='green', label='Promo $100')
for x, y in zip(budgets, avg_churn_100_promo_b):
    axs[1].text(x, y, f'{y:.3f}', ha='right', va='bottom', bbox=dict(facecolor='white', alpha=0.8))
axs[1].set_title('Average Churn Probability vs. Budget - Promo $100')
axs[1].set_ylabel('Average Churn Probability')
axs[1].legend()
axs[1].grid(True)

axs[2].plot(budgets, avg_churn_200_promo_b, marker='o', color='red', label='Promo $200')
for x, y in zip(budgets, avg_churn_200_promo_b):
    axs[2].text(x, y, f'{y:.3f}', ha='right', va='bottom', bbox=dict(facecolor='white', alpha=0.8))
axs[2].set_title('Average Churn Probability vs. Budget - Promo $200')
axs[2].set_xlabel('Budget')
axs[2].set_ylabel('Average Churn Probability')
axs[2].legend()
axs[2].grid(True)

plt.tight_layout()
plt.show()
```


Average Churn Probability vs. Budget - all Promo
No Promo
Average Churn Probability
0.27
0.26
0.25
0.24
0.23
0.22
0.260
0.270
0.230
0.230
0.220


Average Churn Probability vs. Budget - Promo $100
Promo $100
Average Churn Probability
0.25
0.24
0.23
0.22
0.21
0.20
0.240
0.250
0.210
0.210
0.200


Average Churn Probability vs. Budget - Promo $200
Promo $200
Average Churn Probability
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.570
0.540
0.350
0.200
0.270
$100,000
$500,000
$1,000,000
$1,200,000
$1,500,000
Budget