PHASE 6 Decision Optimization: Budget- & Risk-Aware Customer Targeting

```
# STEP 6.1 — Load Phase 5 Artifact (Immutable)

import pandas as pd
import numpy as np

clv_df = pd.read_parquet("phase5_expected_clv.parquet")
print(clv_df.shape)
clv_df.head()
```

```
(5881, 2)
```


```
{"summary":"{\n  \"name\": \"clv_df\",\n  \"rows\": 5881,\n
\"fields\": [\n    {\n      \"column\": \"Customer ID\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1715.4297590182248,\n        \"min\": 12346.0,\n        \"max\":
18287.0,\n        \"num_unique_values\": 5881,\n        \"samples\":
[\n          17776.0,\n          17703.0,\n          12546.0\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"expected_clv\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 13338.156006795914,\n        \"min\":
0.0,\n        \"max\": 401760.5653631753,\n
\"num_unique_values\": 5859,\n        \"samples\": [\n
10381.958679949777,\n          9716.554121162893,\n
2170.4114623373707\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"clv_df"}
```


```
# STEP 6.2 — Define the Decision Problem
```

Decision Problem Formulation

At each decision cycle, the business must select a subset of customers to target with a retention or engagement action.

Each action incurs a cost and is subject to a fixed budget constraint. The objective is to maximize the total expected incremental customer lifetime value (CLV) generated by the chosen actions.

This phase formulates CLV as a decision optimization problem, not merely a predictive metric.

```
# STEP 6.3 — Introduce an Action Space
```

a = 1 → retention offer

a = 0 → do nothing

```
# STEP 6.4 — Define Action Cost & Budget
```

```
# Simple, explainable numbers:
ACTION_COST = 100        # cost per customer
TOTAL_BUDGET = 50000     # total marketing budget

# Maximum customers you can target:
MAX_CUSTOMERS = TOTAL_BUDGET // ACTION_COST
MAX_CUSTOMERS
```

```
500
```

```
# STEP 6.5 — Define Incremental CLV (Key Assumption)

UPLIFT_FACTOR = 0.15   # 15% incremental CLV if targeted

clv_df["incremental_clv"] = (
    UPLIFT_FACTOR * clv_df["expected_clv"]
)

# STEP 6.6 — Risk Control

CAP = clv_df["incremental_clv"].quantile(0.95)

clv_df["incremental_clv_capped"] = (
    clv_df["incremental_clv"].clip(upper=CAP)
)

# STEP 6.7 — Optimization Objective

decision_df = (
    clv_df
    .sort_values("incremental_clv_capped", ascending=False)
    .head(MAX_CUSTOMERS)
    .copy()
)

decision_df["action"] = 1
decision_df.head()
```


```
{"summary":"{\n  \"name\": \"decision_df\",\n  \"rows\": 500,\n
\"fields\": [\n    {\n      \"column\": \"Customer ID\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1762.7971553067346,\n        \"min\": 12346.0,\n        \"max\":
18260.0,\n        \"num_unique_values\": 500,\n        \"samples\": [\
n          15673.0,\n          12472.0,\n          12669.0\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"expected_clv\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 36695.197967126755,\n        \"min\":
13113.531082707117,\n        \"max\": 401760.5653631753,\n
\"num_unique_values\": 500,\n        \"samples\": [\n
16022.275318259515,\n          28045.99450288955,\n
15626.433071094243\n        ],\n        \"semantic_type\": \"\",\n
```

\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"incremental_clv\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 5504.2796950690135,\n   \"min\":
1967.0296624060675,\n   \"max\": 60264.08480447629,\n
\"num_unique_values\": 500,\n   \"samples\": [\n
2403.341297738927,\n   4206.899175433432,\n
2343.9649606641365\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"incremental_clv_capped\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 238.63396731545157,\n
\"min\": 1967.0296624060675,\n   \"max\": 2690.6668241313896,\n
\"num_unique_values\": 206,\n   \"samples\": [\n
2625.194060882544,\n   2661.0084050799755,\n
1976.6313623540116\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"action\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0,\n   \"min\": 1,\n   \"max\": 1,\n
\"num_unique_values\": 1,\n   \"samples\": [\n   1\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe","variable_name":"decision_df"}

```
# STEP 6.8 — Compute Expected Portfolio Gain

total_incremental_value = decision_df["incremental_clv_capped"].sum()
total_cost = len(decision_df) * ACTION_COST

total_incremental_value, total_cost
```

```
(np.float64(1262575.0536835992), 50000)
```

```
# STEP 6.9 — Compare Against Heuristic Baselines

# Baseline 1: Random targeting
random_df = clv_df.sample(MAX_CUSTOMERS, random_state=42)
random_gain = (
    UPLIFT_FACTOR * random_df["expected_clv"]
).sum()

# Baseline 2: Frequency-only targeting (naive)
# Merge frequency from Phase 2
freq_df = (
    pd.read_parquet("phase2_customer_state.parquet")
    [["Customer ID", "frequency"]]
    .drop_duplicates()
)

baseline_df = clv_df.merge(freq_df, on="Customer ID")

freq_top_df = (
    baseline_df
    .sort_values("frequency", ascending=False)
```

```
    .head(MAX_CUSTOMERS)
)

freq_gain = (
    UPLIFT_FACTOR * freq_top_df["expected_clv"]
).sum()

# STEP 6.10 — Decision Comparison Table

comparison_df = pd.DataFrame({
    "Strategy": ["CLV-Optimized", "Frequency-Only", "Random"],
    "Expected Incremental Value": [
        total_incremental_value,
        freq_gain,
        random_gain
    ]
})

comparison_df
```


```
{"summary":"{\n  \"name\": \"comparison_df\",\n  \"rows\": 3,\n
\"fields\": [\n    {\n      \"column\": \"Strategy\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n          \"CLV-
Optimized\",\n          \"Frequency-Only\",\n          \"Random\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Expected Incremental Value\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2031192.289716958,\n        \"min\": 405134.56736144534,\n
\"max\": 4272724.277764421,\n        \"num_unique_values\": 3,\n
\"samples\": [\n          1262575.0536835992,\n
4272724.277764421,\n          405134.56736144534\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"comparison_df"}
```