04-decision-engine

January 16, 2026

1 Purpose of the Decision Engine

1.1 Executive Decision Engine

This notebook converts analytical outputs into business decisions by: - Translating forecasts into revenue scenarios - Quantifying financial risk - Recommending segment-wise actions - Estimating ROI for leadership planning

The goal is not prediction accuracy alone, but **decision confidence**.

Load Required Outputs

[1]:
```
import pandas as pd
import numpy as np

forecast_df = pd.read_csv(
    r"D:
 ↪\decision-intelligence-project\Data\processed\revenue_forecast_scenarios.
 ↪csv",
    index_col=0,
    parse_dates=True
)

segment_summary = pd.read_csv(
    r"D:\decision-intelligence-project\Notebook\segment_decision_summary.csv"
)

forecast_df.head(), segment_summary.head()
```

[1]:
```
(            Base_Forecast       Lower_CI      Upper_CI      Best_Case  \
 2011-07-01  735969.970110  393240.409229  1.078700e+06  809566.967121
 2011-08-01  738507.933895  179972.174243  1.297044e+06  812358.727284
 2011-09-01  738257.981326   32877.666870  1.443638e+06  812083.779459
 2011-10-01  738282.598024  -88775.027676  1.565340e+06  812110.857826
 2011-11-01  738280.173636 -194673.231200  1.671234e+06  812108.191000

                Worst_Case
 2011-07-01  662372.973099
 2011-08-01  664657.140505
```

```
2011-09-01  664432.183193
2011-10-01  664454.338221
2011-11-01  664452.156273  ,
   Cluster  Avg_Recency  Avg_Frequency   Avg_Monetary  Customer_Count  \
0        0    43.702685       3.682711    1359.049284            3054
1        1   248.075914       1.552015     480.617480            1067
2        2     7.384615      82.538462  127338.313846              13
3        3    15.500000      22.333333   12709.090490             204

             Decision_Action
0        Exit / Deprioritize
1        Exit / Deprioritize
2  Defend (High Value Loyal)
3  Defend (High Value Loyal)  )
```

Revenue Impact Analysis

Aggregate 6-Month Outlook

[2]:
```
revenue_summary = {
    "Worst_Case_Revenue": forecast_df["Worst_Case"].sum(),
    "Base_Case_Revenue": forecast_df["Base_Forecast"].sum(),
    "Best_Case_Revenue": forecast_df["Best_Case"].sum()
}

revenue_summary_df = pd.DataFrame.from_dict(
    revenue_summary, orient="index", columns=["6_Month_Revenue"]
)

revenue_summary_df
```

[2]:
```
                    6_Month_Revenue
Worst_Case_Revenue     3.984821e+06
Base_Case_Revenue      4.427579e+06
Best_Case_Revenue      4.870337e+06
```

1.1.1 Insight:

- Revenue variability between worst and best case indicates strategic risk exposure.
- Leadership can plan budgets using base case and retain contingency buffers.

Risk Exposure Quantification

[3]:
```
risk_df = forecast_df.copy()

risk_df["Downside_Risk"] = (
    risk_df["Base_Forecast"] - risk_df["Worst_Case"]
)
```

```
risk_df["Upside_Potential"] = (
    risk_df["Best_Case"] - risk_df["Base_Forecast"]
)

risk_df[["Downside_Risk", "Upside_Potential"]].describe()
```

[3]:

	Downside_Risk	Upside_Potential
count	6.000000	6.000000
mean	73792.984490	73792.984490
std	96.467944	96.467944
min	73596.997011	73596.997011
25%	73826.352940	73826.352940
50%	73828.029302	73828.029302
75%	73828.205162	73828.205162
max	73850.793389	73850.793389

- Downside risk defines capital protection requirements
- Upside potential defines growth investment opportunity

Segment-Wise Decision Mapping

[4]:
```
segment_summary.columns
```

[4]:
```
Index(['Cluster', 'Avg_Recency', 'Avg_Frequency', 'Avg_Monetary',
       'Customer_Count', 'Decision_Action'],
      dtype='object')
```

[6]:
```
cluster_to_segment = {
    0: "Low Value",
    1: "High Value",
    2: "Mid Value",
    3: "Churn Risk"
}

segment_summary["Segment"] = segment_summary["Cluster"].map(cluster_to_segment)
```

[7]:
```
segment_actions = {
    "High Value": "Retention & Loyalty Programs",
    "Mid Value": "Cross-Sell & Upsell Campaigns",
    "Low Value": "Cost-Control & Automation",
    "Churn Risk": "Win-Back Offers"
}

segment_summary["Recommended_Action"] = (
    segment_summary["Segment"].map(segment_actions)
)
```

[9]:
```
segment_summary["Recommended_Action"] = (
    segment_summary["Cluster"].map(segment_actions)
)
```

ROI Simulation (THIS IS GOLD)

Assumptions (Clearly Stated)

[10]:
```
roi_assumptions = {
    "High Value": {"investment": 50000, "expected_uplift": 0.12},
    "Mid Value": {"investment": 30000, "expected_uplift": 0.08},
    "Low Value": {"investment": 15000, "expected_uplift": 0.03},
    "Churn Risk": {"investment": 20000, "expected_uplift": 0.06}
}
```

Segment Revenue

[12]:
```
segment_summary["Estimated_Revenue"] = (
    segment_summary["Avg_Monetary"] *
    segment_summary["Avg_Frequency"] *
    segment_summary["Customer_Count"]
)
```

ROI Calculation

[13]:
```
roi_results = []

for _, row in segment_summary.iterrows():
    segment = row["Segment"]
    base_revenue = row["Estimated_Revenue"]

    invest = roi_assumptions[segment]["investment"]
    uplift = roi_assumptions[segment]["expected_uplift"]

    projected_gain = base_revenue * uplift
    roi = (projected_gain - invest) / invest

    roi_results.append({
        "Segment": segment,
        "Base_Revenue": base_revenue,
        "Investment": invest,
        "Projected_Gain": projected_gain,
        "ROI": roi
    })

roi_df = pd.DataFrame(roi_results)
roi_df
```

[13]:

	Segment	Base_Revenue	Investment	Projected_Gain	ROI
0	Low Value	1.528523e+07	15000	4.585568e+05	29.570455
1	High Value	7.959025e+05	50000	9.550831e+04	0.910166
2	Mid Value	1.366340e+08	30000	1.093072e+07	363.357362
3	Churn Risk	5.790262e+07	20000	3.474157e+06	172.707849

Final Executive Recommendation

1.2 Executive Recommendations

1. Prioritize investment in **High Value** segments due to highest ROI.
2. Maintain defensive strategies for **Churn Risk** customers.
3. Limit spending on **Low Value** segments to cost efficiency initiatives.
4. Use Base Case revenue forecast for budgeting and Worst Case for risk buffers.

This decision framework enables data-backed, risk-aware leadership planning.

[14]:
```
# Save ROI simulation output for downstream risk analysis

roi_df.to_csv(
    r"D:
 ↪\decision-intelligence-project\Data\Processed_Data\roi_simulation_results.
 ↪csv",
    index=False
)

print("roi_simulation_results.csv saved successfully")
```

```
roi_simulation_results.csv saved successfully
```

[ ]:
```

```