PHASE 4 Hazard Modeling: Time-Dependent Churn Risk

```
# STEP 4.1 — Load Phase 3 Artifact (Immutable)

import pandas as pd

df = pd.read_parquet("phase3_state_with_survival.parquet")
print(df.shape)
df.head()
```

```
(37039, 10)
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 37039,\n  \"fields\":
[\n    {\n      \"column\": \"Customer ID\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 1721.1264193434051,\
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\": \"InvoiceDate\",\n
\"properties\": {\n        \"dtype\": \"date\",\n        \"min\":
\"2009-12-01 07:45:00\",\n        \"max\": \"2011-12-09 12:50:00\",\n
\"num_unique_values\": 34591,\n        \"samples\": [\n
\"2010-09-20 17:32:00\",\n          \"2011-04-15 08:45:00\",\n
\"2010-01-20 09:50:00\"\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"recency_days\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 75.7097797248008,\n
\"min\": 0.0,\n        \"max\": 714.0,\n        \"num_unique_values\":
515,\n        \"samples\": [\n          384.0,\n          682.0,\n
690.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"frequency\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 37,\n        \"min\": 0,\n
\"max\": 399,\n        \"num_unique_values\": 400,\n
\"samples\": [\n          209,\n          280,\n          33\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"monetary_avg\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 777.8735726422763,\n        \"min\":
0.0,\n        \"max\": 84236.25,\n        \"num_unique_values\":
35842,\n        \"samples\": [\n          261.8607692307692,\n
443.9271428571429,\n          339.4130612244898\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"delta_revenue\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1635.9432289056133,\n        \"min\": -42409.1,\n        \"max\":
168466.7,\n        \"num_unique_values\": 28759,\n        \"samples\":
[\n          159.18,\n          80.27999999999997,\n          -
516.1199999999999\n        ],\n        \"semantic_type\": \"\",\n
```

\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"delta_recency\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 75.6244236542911,\n   \"min\": -
618.0,\n   \"max\": 701.0,\n   \"num_unique_values\": 796,\n
\"samples\": [\n   387.0,\n   492.0,\n   -166.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"is_alive\",\n
\"properties\": {\n   \"dtype\": \"boolean\",\n
\"num_unique_values\": 2,\n   \"samples\": [\n   true,\n
false\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"duration\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 145,\n   \"min\": 0,\n
\"max\": 738,\n   \"num_unique_values\": 676,\n
\"samples\": [\n   586,\n   492\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"event\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\":
0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n
\"samples\": [\n   0,\n   1\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe","variable_name":"df"}

```
# STEP 4.2 — Choose Time Representation
```

STEP 4.2 — Choice of Time Representation

In this phase, we model churn risk using a **discrete-time survival framework**.

The continuous survival duration (measured as days since last observed transaction) is discretized into fixed-length time intervals (monthly bins).

This choice is motivated by:

1. Interpretability:
   Discrete-time hazard directly represents the probability of churn in the next time interval, conditional on survival so far.

2. Data characteristics:
   Transaction events occur at irregular time gaps, making discrete-time modeling more appropriate than continuous-time assumptions.

3. Practical and academic validity:
   Discrete-time survival analysis is a standard and accepted approach in non-contractual customer lifetime modeling and decision science literature.

This representation enables a clear person-period dataset construction and supports interpretable hazard estimation using generalized linear models.

```
# STEP 4.3 — Define Time Bins
```

```
df["time_bin"] = (df["duration"] // 30).astype(int)
df["time_bin"].describe()
```

```
count    37039.000000
mean         2.609439
std          4.806882
min          0.000000
25%          0.000000
50%          0.000000
75%          2.000000
max         24.000000
Name: time_bin, dtype: float64
```

```
# STEP 4.4 — Expand to Person-Period Format

rows = []

for _, row in df.iterrows():
    for t in range(row["time_bin"] + 1):
        rows.append({
            "Customer ID": row["Customer ID"],
            "time_bin": t,
            "event": int((t == row["time_bin"]) and (row["event"] ==
1)),
            "recency_days": row["recency_days"],
            "frequency": row["frequency"],
            "monetary_avg": row["monetary_avg"],
            "delta_revenue": row["delta_revenue"],
            "delta_recency": row["delta_recency"]
        })

person_period_df = pd.DataFrame(rows)
person_period_df.head()
```

{"type":"dataframe","variable_name":"person_period_df"}

```
# STEP 4.5 — Sanity Checks

# Event happens once per customer max
person_period_df.groupby("Customer ID")["event"].sum().max()
```

```
155
```

```
# Hazard base rate by time
person_period_df.groupby("time_bin")["event"].mean().head(10)
```

```
time_bin
0    0.000000
1    0.000000
2    0.000000
3    0.000000
```

```
4    0.000000
5    0.000000
6    0.099035
7    0.099374
8    0.087319
9    0.081105
Name: event, dtype: float64
```

```
# STEP 4.5.1 — Missing Value Handling
```

```
# STEP 4.5.1 — Handle Missing Values (CLV-safe)

from sklearn.impute import SimpleImputer

features = [
    "recency_days",
    "frequency",
    "monetary_avg",
    "delta_revenue",
    "delta_recency",
    "time_bin"
]

X = person_period_df[features]
y = person_period_df["event"]

# Median imputation for numerical stability
imputer = SimpleImputer(strategy="median")
X_imputed = imputer.fit_transform(X)

# Logistic Regression

from sklearn.linear_model import LogisticRegression

model = LogisticRegression(max_iter=1000)
model.fit(X_imputed, y)
```

```
LogisticRegression(max_iter=1000)
```

```
# Quick sanity check
import numpy as np

np.isnan(X_imputed).sum()
```

```
np.int64(0)
```

Missing Value Handling

Certain customer state variables (e.g., recency and behavioral deltas) are undefined for early customer events, resulting in missing values.

We apply median imputation to numerical features prior to hazard modeling. This choice preserves sample size, avoids survival bias, and maintains interpretability of the discrete-time hazard model.

```
# STEP 4.6 — Fit a Simple Hazard Model (Interpretable)



from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression

features = [
    "recency_days",
    "frequency",
    "monetary_avg",
    "delta_revenue",
    "delta_recency",
    "time_bin"
]

X = person_period_df[features]
y = person_period_df["event"]

# Median imputation (CLV-safe)
imputer = SimpleImputer(strategy="median")
X_imputed = imputer.fit_transform(X)

# Discrete-time hazard model
hazard_model = LogisticRegression(max_iter=1000)
hazard_model.fit(X_imputed, y)
```

```
LogisticRegression(max_iter=1000)
```

We estimate the discrete-time hazard function using logistic regression, where the model predicts the probability of churn in the next time interval conditional on survival up to that interval.

Logistic regression is chosen for its interpretability and stability, allowing direct inspection of how customer state variables influence churn risk over time.

```
# STEP 4.7 — Inspect Hazard Direction (VERY IMPORTANT)

coef_df = pd.DataFrame({
    "feature": features,
    "coef": hazard_model.coef_[0]
}).sort_values("coef")

coef_df
```

{"summary":"{\n \"name\": \"coef_df\",\n \"rows\": 6,\n \"fields\":
[\n   {\n   \"column\": \"feature\",\n   \"properties\": {\n
\"dtype\": \"string\",\n   \"num_unique_values\": 6,\n
\"samples\": [\n   \"delta_recency\",\n
\"monetary_avg\",\n   \"time_bin\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"coef\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 0.10461135381789329,\n
\"min\": -0.0021227835154468255,\n   \"max\":
0.25783210818024344,\n   \"num_unique_values\": 6,\n
\"samples\": [\n   -0.0021227835154468255,\n   -
1.4893385790788094e-05,\n   0.25783210818024344\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe","variable_name":"coef_df"}

```
hasattr(hazard_model, "coef_")
```

```
True
```

```
# STEP 4.8 — Save Phase 4 Artifact

person_period_df.to_parquet(
    "phase4_person_period_dataset.parquet",
    index=False
)
```