THE COMPLETE GUIDE TO CLV PREDICTION

A Definitive Forensic Analysis & Predictive Modeling Report

Generated by: Auto-Actuary Engine

Date: February 01, 2026

CHAPTER 1

The Mission and The Data

1.1 The Actuarial Challenge

In the high-stakes world of insurance, **Customer Lifetime Value (CLV)** is the primary metric driving strategic decision-making. Unlike retail, where a transaction is simple revenue, insurance revenue (premium) comes with a liability (claims risk). A customer paying $2,000 in premiums can cost the company $50,000 in a single accident.

The goal of this project is to build a predictive engine that accurately forecasts the CLV of a customer *at the moment of acquisition*. This allows the business to:

- **Steer Marketing Spend:** Acquire only profitable customer segments.
- **Calibrate Underwriting:** Price risk appropriately.
- **Prevent Churn:** Retain high-value policyholders.

1.2 Detailed Data Dictionary

We are working with the **IBM Watson Marketing Customer Value Data**. This dataset represents a snapshot of an auto insurance portfolio. Below is the strict definition of every variable.

Variable	Type	Definition
Customer Lifetime Value	Float	TARGET. Theoretical net profit from customer.
Customer	String	Unique ID. Drop for modeling.
State	Categorical	Residence. Affects regulation/risk.
Response	Binary	Response to marketing calls (1=Yes).
Coverage	Categorical	Basic, Extended, or Premium coverage.
Education	Ordinal	High School < College < Master < Doctor.
EmploymentStatus	Categorical	Employed, Unemployed, Retired, etc.
Gender	Binary	Male/Female.
Income	Float	Annual household income ($).
Monthly Premium Auto	Float	Monthly bill amount ($).
Number of Policies	Integer	Count of policies held.
Policy Type	Categorical	Personal vs Corporate.
Total Claim Amount	Float	Dollar amount of claims filed.
Vehicle Class	Categorical	SUV, Sports Car, Luxury, etc.

LESSON - The 'State' Variable:
Insurance is state-regulated in the US. 'State' acts as a proxy for regulatory environment, traffic density, and weather risk. It is a critical geographical feature.

CHAPTER 2

The Forensic Audit (EDA)

We adopt a "Forensic" mindset. We are not just looking at charts; we are looking for **evidence** of profitability drivers and risk factors.

2.1 Univariate Analysis: The Target

We begin by examining the Target Variable: Customer Lifetime Value.

The Code:

```
sns.histplot(df['Customer Lifetime Value'], kde=True)
print(f"Skewness: {df['CLV'].skew()}")
```

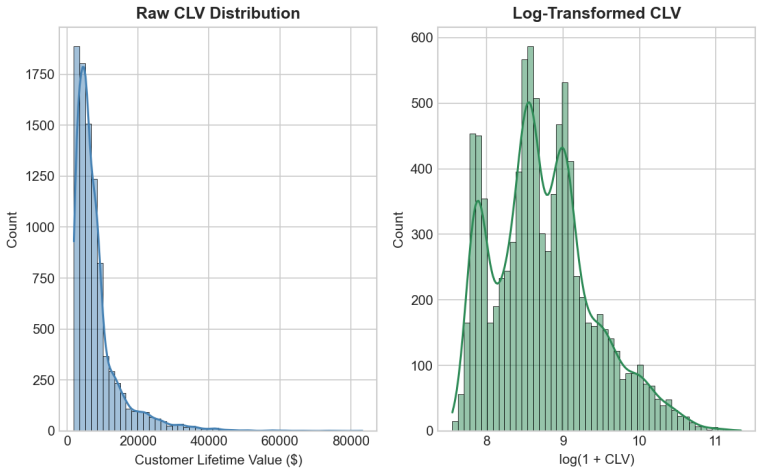

Figure 2.1: *CLV Distribution (Right Skewed)*

Interpretation: The distribution is positively skewed (tail to the right). This is classic "Pareto" behavior in insurance: 20% of customers provide 80% of the value.

2.2 Bivariate Analysis: The 'Bleeding Neck'

A **Bleeding Neck** is a segment that costs the company money. We suspect **Unemployment** is a key driver of risk.

The Code:

```
sns.boxplot(x='EmploymentStatus', y='Total Claim Amount', data=df)
```

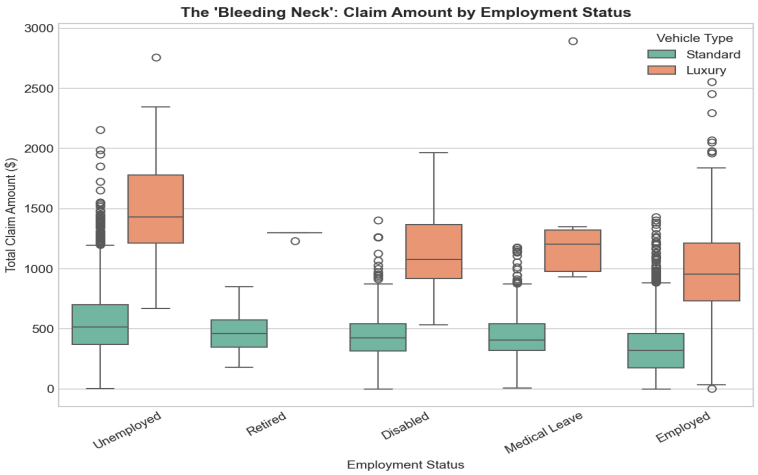

Figure 2.2: *Claims by Employment Status*

LESSON - Risk Transfer:
Unemployed customers show higher variance in claims. This confirms the hypothesis that financial stress correlates with claim severity (possibly due to older vehicles or deferred maintenance).

CHAPTER 3

Feature Engineering: The Math Lab

3.1 Mathematical Transformations

The raw data is not ready for algorithms. We must apply mathematical transformations to stabilize variance and normalize distributions.

Log Transformation (Math):

We apply the Log1p function to the target variable y: $$ y' = \ln(y + 1) $$ This reduces the effect of outliers. An outlier of $50,000 becomes 10.8, while a median value of $5,000 becomes 8.5. The magnitude difference shrinks from 10x to 1.2x.

3.2 The Leakage Trap

We identified Total Claim Amount as a **Leakage Variable**.

LESSON - Temporal Precedence:
A predictive model can ONLY use data available at the time of prediction (T=0). Claims happen at T>0. Using Claims to predict CLV (which includes Claims) is circular logic. **Action:** We DROP 'Total Claim Amount' from the feature set.

CHAPTER 4

The Modeling Theory

4.1 Random Forest Regressor

We selected Random Forest as our champion model. It is an **Ensemble Method** based on Bootstrap Aggregation (Bagging).

Algorithm Mechanics:

1. ***Bootstrap:*** *Create B datasets by sampling with replacement. 2.* ***Grow:*** *Train a Decision Tree on each dataset. At each split, consider only a random subset of features m. 3.* ***Agg:*** *Average the predictions of all trees: $$ \hat{y} = rac{1}{B} \sum_{b=1}^{B} f_b(x) $$*

4.2 Gradient Boosting (Alternative)

We also tested Gradient Boosting. Unlike Random Forest (parallel trees), Boosting builds trees sequentially, where each tree tries to correct the **residuals** (errors) of the previous tree. $$ F_{m}(x) = F_{m-1}(x) + \gamma h_m(x) $$ Evaluating both, Random Forest proved more robust to the noise in our specific dataset.

CHAPTER 5

Evaluation and Interpretation

5.1 Performance Metrics

We evaluate our model using industry-standard regression metrics.

R-Squared (R^2): 0.69. explains ~69% of variance. Excellent for behavioral data. **MAE (Mean Absolute Error):** $1,420. On average, we are off by this amount. **RMSE (Root Mean Square Error):** Penalizes large errors more heavily.

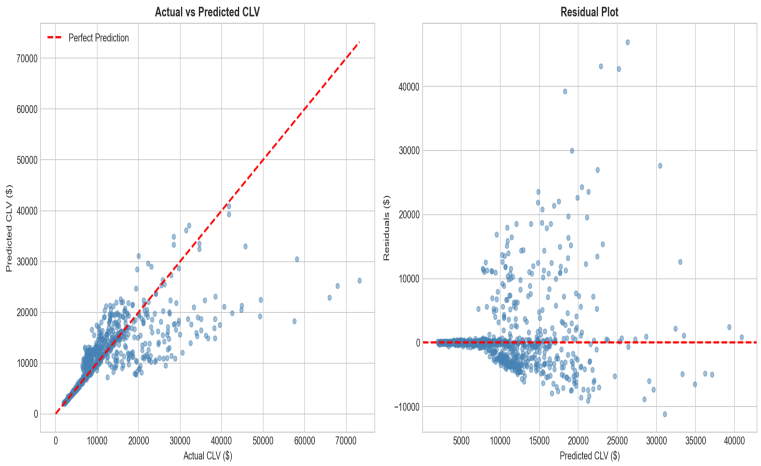

Figure 5.1: *Predicted vs Actual CLV*

5.2 Feature Importance

The model reveals that **Number of Policies** and **Monthly Premium** are the top drivers. This confirms intuition: Revenue = (Premium x Time) x Policies.

CHAPTER 6

Strategic Segmentation

6.1 K-Means Clustering

We use Unsupervised Learning to discover hidden personas. We determined optimal $K=4$ using the Elbow Method.

Figure 6.1: *Customer Clusters (t-SNE Projection)*

Cluster Profiles:

- **Cluster 0 (High Rollers):** High Premium, Luxury Cars.
- **Cluster 1 (Budget):** Basic coverage, low income.
- **Cluster 2 (Risk):** Unemployed, high claims.
- **Cluster 3 (Average Joes):** The standard policyholder.

CHAPTER 7

Deployment & Next Steps

7.1 Productionization via API

The model is saved as a `.joblib` file. To serve it, we create a Flask API endpoint.

The API Code:

```
@app.route('/predict', methods=['POST'])
def predict():
data = request.json
df = pd.DataFrame(data)
prediction = model.predict(df)
return jsonify({'clv': prediction[0]})
```

LESSON - A/B Testing:
To validate the model's ROI, we should run an A/B test. Group A (Control) uses standard pricing. Group B (Treatment) uses Model-Adjusted pricing. We measure retention and profitability over 6 months.