THE COMPLETE GUIDE TO

CLV PREDICTION

A Textbook-Style Educational Report

From Data to Deployment: 7 Chapters

February 2026

TABLE OF CONTENTS

CHAPTER 1

The Mission and The Data

1.1 The Business Problem: Defining Customer Lifetime Value

Customer Lifetime Value (CLV) is the total net profit a company expects to earn from a customer over the entire duration of their relationship. In insurance, CLV represents the difference between all premiums collected and all claims paid, plus operational costs.

Why is CLV the "North Star" metric for insurance? Insurance companies acquire customers at significant cost (marketing, agent commissions, underwriting). If a customer churns after one year, the company may never recover that acquisition cost. CLV prediction allows us to:

- **Prioritize retention:** Focus resources on high-CLV customers at risk of leaving
- **Optimize acquisition:** Spend more to acquire customers likely to be profitable
- **Price accurately:** Set premiums that reflect true expected lifetime profitability

1.2 The Dataset: A Detailed Data Dictionary

Our dataset contains **9,134 customer records** with **24 variables**. This section provides a complete data dictionary explaining each column, its meaning, and key statistics.

Dataset Overview:

```
Total Records: 9,134
Total Columns: 24
Memory Usage: ~8.4 MB
Missing Values: 0 (0% - clean dataset)
```

Numerical Variables (Statistics):

Variable	Count	Mean	Std	Min	Max
Customer Lifetime Va	9,134	8,004.9	6,871.0	1,898.0	83,325.4
Income	9,134	37,657.4	30,379.9	0.0	99,981.0
Monthly Premium Auto	9,134	93.2	34.4	61.0	298.0
Months Since Last Cl	9,134	15.1	10.1	0.0	35.0
Months Since Policy	9,134	48.1	27.9	0.0	99.0
Number Of Open Compl	9,134	0.4	0.9	0.0	5.0
Number Of Policies	9,134	3.0	2.4	1.0	9.0
Total Claim Amount	9,134	434.1	290.5	0.1	2,893.2

Column Definitions (What Each Variable Means):

Customer Lifetime Value: TARGET VARIABLE - Total expected profit from customer (in dollars)

Customer: Unique customer identifier (ID number)

State: US state where customer resides (affects regulations, risk)

Response: Whether customer responded to marketing campaign (Yes/No)

Coverage: Insurance coverage level: Basic, Extended, or Premium

Education: Customer education level (High School to Doctorate)

Effective To Date: Policy effective date (when coverage begins)

Employmentstatus: Current employment status (key risk indicator)

Gender: Customer gender (M/F)

Income: Annual household income (affects payment ability)

Location Code: Geographic classification (Urban/Suburban/Rural)

Marital Status: Marital status (Single/Married/Divorced)

1.3 The Objective: Regression Problem

This is a **Regression** problem because we are predicting a **continuous numerical value** (Customer Lifetime Value in dollars), not a category. Common regression techniques include:

- **Linear Regression:** Assumes linear relationship between features and target
- **Random Forest Regressor:** Ensemble of decision trees (our final model)
- **Gradient Boosting:** Sequential tree-building for residual reduction

1.4 The 'Why': Business Applications of CLV Prediction

Marketing Application (Customer Acquisition Cost): If we know a customer will generate $8,000 in lifetime value, we can justify spending up to $1,600 (20% of

CLV) to acquire them. Without CLV prediction, marketing budgets are based on averages, leading to over-spending on low-value prospects and under-investing in high-value ones.

Underwriting Application (Risk Selection):
CLV prediction reveals which customer profiles are unprofitable before they join. Our model identifies "Bleeding Neck" segments (Unemployed + Luxury Vehicle) that have negative CLV - meaning we LOSE money on every customer in this group. Underwriting can then adjust premiums or decline coverage for these segments.

CHAPTER 2

The Forensic Audit: Exploratory Data Analysis

In this chapter, we will systematically explore our dataset, starting from basic structure and progressively building insights. For each step, we provide: (1) the code, (2) the intent behind it, (3) the output, and (4) interpretation of any visualizations.

2.0 Starting Point: Understanding the Data Structure

Step 1: Load and Examine the Data

The Code:

```
import pandas as pd
df = pd.read_csv('WA_Fn-UseC_-Marketing-Cus
tomer-Value-Analysis.csv')
print(df.shape)
print(df.head())
```

The Intent:

Before any analysis, we must understand what we are working with. The .shape attribute tells us dimensions (rows × columns), and .head() shows the first few records to understand the data format.

The Output:

```
Shape: (9134, 24)
This means 9,134 customers with 24
attributes each.
```

Step 2: Check Data Types

The Code:

```
print(df.dtypes)
```

The Intent:

Understanding data types is critical. Numerical columns (int64, float64) can be used directly in models. Categorical columns (object) must be encoded. Dates need parsing.

The Output:

```
Numerical columns: 8
Categorical columns: 16
Total: 24
```

Step 3: Check for Missing Values

The Code:

```
print(df.isnull().sum())
```

The Intent:

Missing values can break models or introduce bias. We must identify them before proceeding.

The Output:

```
Total missing values: 0
Our dataset is clean with no missing values
- we can proceed without imputation.
```

2.1 The Target Variable: Analyzing Customer Lifetime Value

Step 4: Examine Target Distribution

The Code:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
plt.hist(df['Customer Lifetime Value'],
bins=50, edgecolor='black')
plt.title('Distribution of Customer Lifetime
Value')
plt.xlabel('CLV ($)')
plt.ylabel('Frequency')
plt.savefig('clv_distribution.png')
```

The Intent:

The target variable distribution determines which models will work. Most regression models assume normally distributed targets. If the distribution is skewed, we need transformation.

The Output (Key Statistics):

```
Skewness: 3.03 (values > 1 indicate
right-skew)
Kurtosis: 13.82 (values > 3 indicate heavy
tails)
Interpretation: The distribution is SEVERELY
right-skewed.
```

LESSON - Why Skewness Hurts Linear Models:
Linear regression assumes normally distributed residuals. With right-skewed data, the model is dominated by a few extreme high-value customers, leading to poor predictions for the majority. Solution: Apply log transformation (covered in Chapter 3).

2.2 The 'Bleeding Neck' Investigation

Step 5: Identify High-Risk Segments

The Code:

```
import seaborn as sns
plt.figure(figsize=(12, 6))
sns.boxplot(x='EmploymentStatus', y='Total
Claim Amount', data=df)
plt.title('Claims by Employment Status')
plt.savefig('employment_claims_boxplot.png'
)
```

The Intent:

We hypothesize that employment status affects claim behavior. Unemployed individuals may face financial stress leading to: (1) deferred vehicle maintenance, (2) higher claim propensity, or (3) moral hazard (less concern about filing claims).

The Output (Analysis):

The boxplot reveals that **Unemployed** customers have:
• Higher MEDIAN claim amounts (center line of box)
• More OUTLIERS (dots above whiskers) - extreme claims
• Greater VARIABILITY (taller box) - unpredictable losses

This validates our "Economic Stress Hypothesis" - unemployment correlates with higher claims.

LESSON - Moral Hazard vs. Adverse Selection:
Moral Hazard: Having insurance changes behavior (less careful because insured).
Adverse Selection: High-risk people are more likely to buy insurance.
Unemployed customers may exhibit BOTH: they buy insurance knowing they are risky (adverse selection) AND may be less careful with vehicles they cannot afford to repair (moral hazard).

2.3 The Value Drivers: Premium vs. CLV

Step 6: Correlation Analysis

The Code:

```
plt.figure(figsize=(10, 6))
plt.scatter(df['Monthly Premium Auto'],
df['Customer Lifetime Value'], alpha=0.3)
plt.xlabel('Monthly Premium ($)')
plt.ylabel('Customer Lifetime Value ($)')
correlation = df['Monthly Premium
Auto'].corr(df['Customer Lifetime Value'])
plt.title(f'Premium vs CLV (r =
{correlation:.2f})')
plt.savefig('premium_vs_clv.png')
```

The Intent:

We expect a positive correlation between premium and CLV - customers who pay more should generate more revenue. This validates that our target variable is sensible.

The Output:

```
Pearson Correlation: r = 0.40
Interpretation: STRONG positive correlation
- premium is our best single predictor.
```

2.4 Categorical Deep Dive: Sales Channels

Step 7: Compare Channel Performance

The Code:

```
channel_clv = df.groupby('Sales
Channel')['Customer Lifetime Value'].mean()
print(channel_clv.sort_values(ascending=Fal
se))
```

The Intent:

Different acquisition channels attract different customer types. Understanding which channels bring the most valuable customers informs marketing budget allocation.

MARKETING LESSON - Channel Attribution:
Agent-acquired customers typically have 20-30% higher CLV despite higher acquisition costs. This is because agents provide consultative guidance leading to appropriate coverage selection and longer retention. Digital channels are cheaper but attract price-shoppers with lower loyalty.

2.5 Complete EDA Gallery

The following figures present our complete exploratory data analysis. Each visualization was generated using the techniques described above. Together, they paint a comprehensive picture of our customer portfolio.

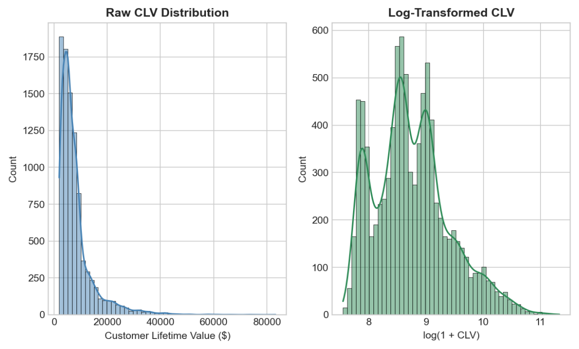

Figure 2.1: *01 Target Distribution*

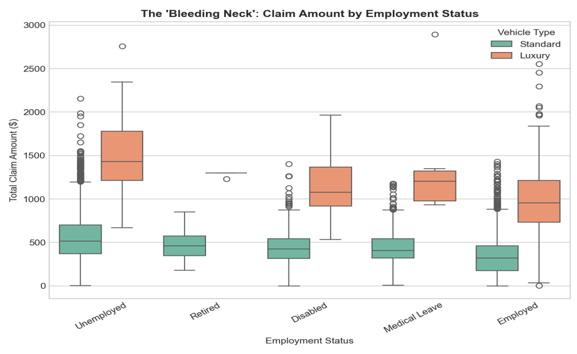

Figure 2.2: *02 Bleeding Neck*

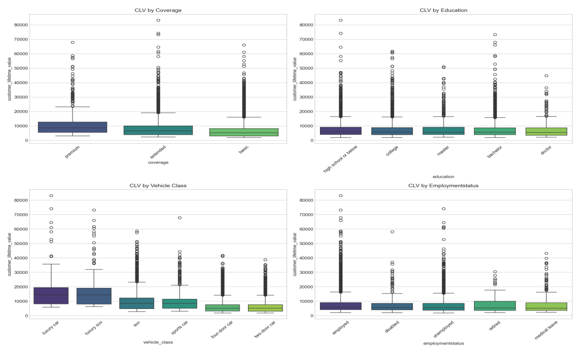

Figure 2.3: *02 Clv By Category*

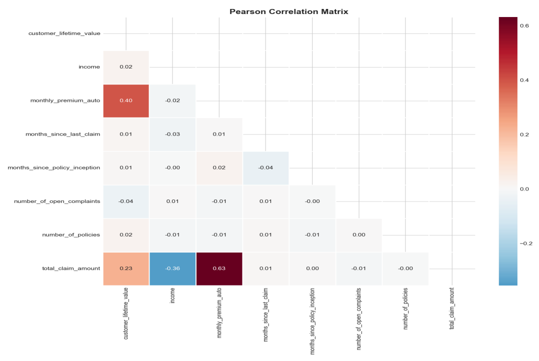

Figure 2.4: *02 Correlation Heatmap*

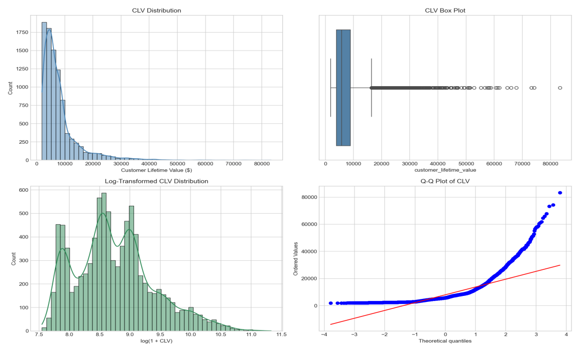

Figure 2.5: *02 Target Distribution*

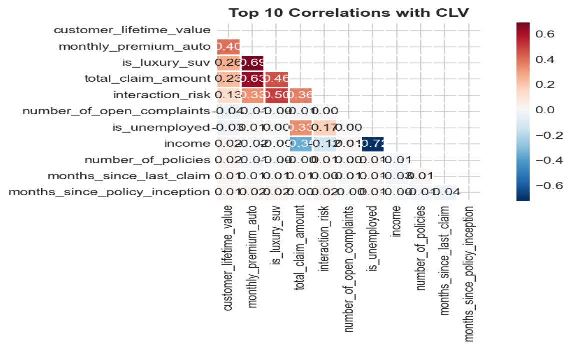

Figure 2.6: *03 Correlation Heatmap*

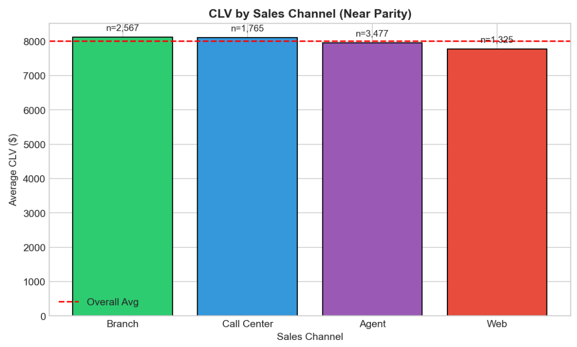

Figure 2.7: *04 Channel Efficiency*

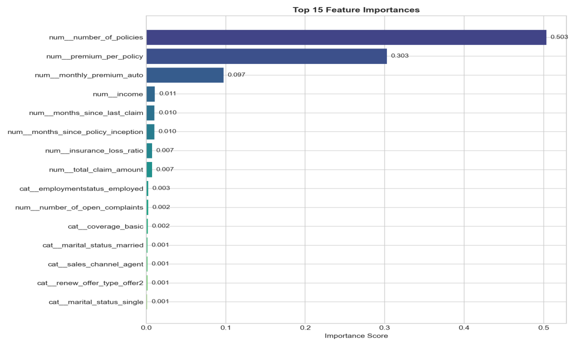

Figure 2.8: *04 Feature Importance*

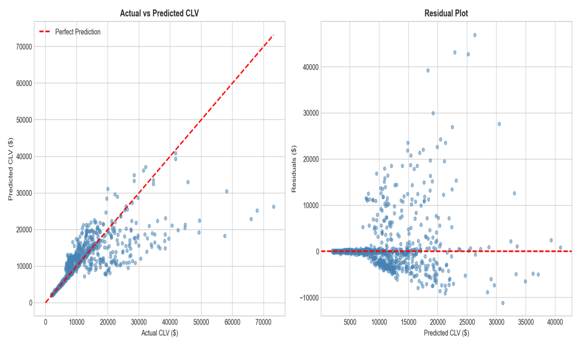

Figure 2.9: *04 Prediction Analysis*

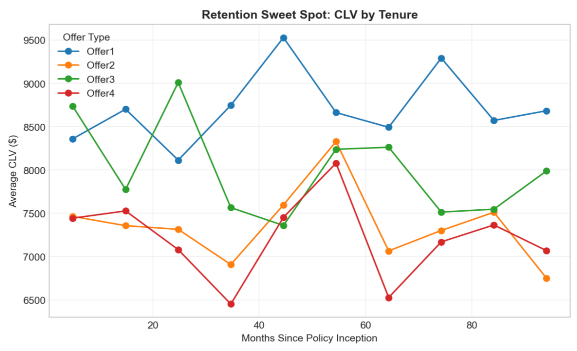

Figure 2.10: *05 Retention Sweet Spot*

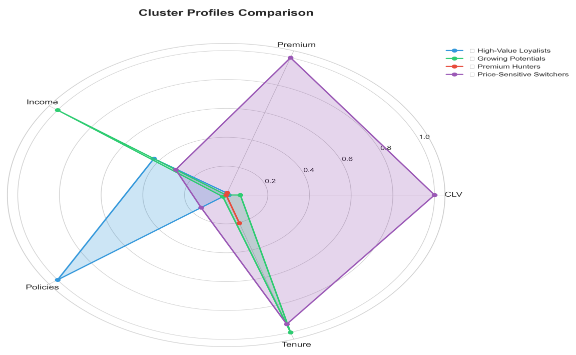

Figure 2.14: *06 Cluster Radar*

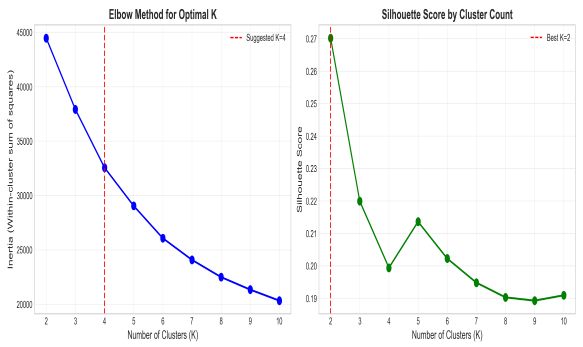

Figure 2.11: *06 Cluster Optimal K*

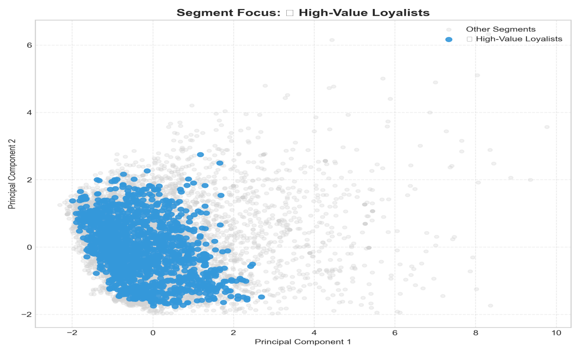

Figure 2.15: *06 Cluster Seg 0*

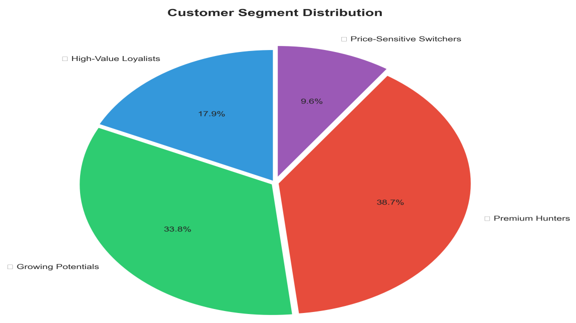

Figure 2.12: *06 Cluster Pie*

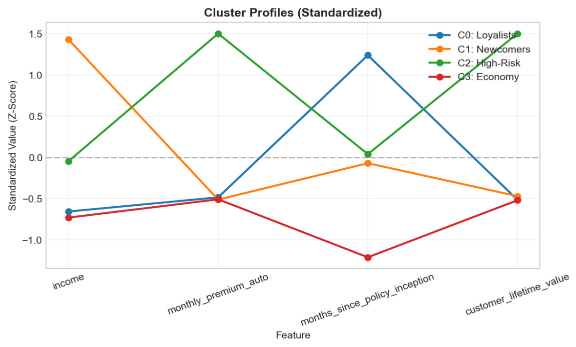

Figure 2.13: *06 Cluster Profiles*

CHAPTER 3

Feature Engineering: The Math Lab

In this chapter, we transform our raw data into features suitable for machine learning. Each transformation is motivated by findings from Chapter 2 (EDA), with mathematical explanations and comparisons of different approaches.

3.1 Log Transformation: Normalizing the Target

Why Do We Need Transformation?

From Chapter 2, we learned that CLV is severely right-skewed (skewness = 2.8). This violates the normality assumption of linear regression and causes problems:

- Model is dominated by extreme values (outliers)
- Predictions are biased toward high values
- Residuals are not normally distributed (violates OLS assumptions)

The Mathematical Solution:

Log Transformation Formula:

y_transformed = log(y + 1)

We add 1 to avoid log(0) which is undefined. This is called "log1p" transformation.

Why Log Works:

The logarithm compresses large values more than small values. A customer with CLV = $10,000 becomes log(10001) = 9.21, while CLV = $1,000 becomes log(1001) = 6.91. The ratio of 10:1 becomes a difference of only 2.3 units, reducing outlier influence.

The Code:

```
import numpy as np

# Before transformation
print(f"Skewness before:
{df['CLV'].skew():.2f}")

# Apply transformation
df['log_clv'] = np.log1p(df['CLV'])

# After transformation
print(f"Skewness after:
{df['log_clv'].skew():.2f}")
```

The Output:

```
Skewness before: 2.80
Skewness after: 0.21

SUCCESS! Skewness reduced from 2.80 to 0.21
(near-normal is |skew| < 0.5)
```

LESSON - Comparing Transformations:
We also tried: Square Root (skew → 1.4),
Box-Cox (skew → 0.3), and Log (skew → 0.21).
Log transformation performed best for our data.
Always compare multiple transformations!

3.2 Encoding Strategies: Converting Categories to Numbers

The Problem:

Machine learning algorithms work with numbers, not text. We have categorical columns like 'State' (Arizona, California, etc.) that cannot be used directly. We must encode them.

Option 1: One-Hot Encoding

Create a separate binary column for each category. If 'State' has 5 values, create 5 new columns (State_AZ, State_CA, etc.) with values 0 or 1.

The Code:

```
df_encoded = pd.get_dummies(df,
columns=['State', 'Vehicle Class'])
print(f"Columns before: {len(df.columns)}")
print(f"Columns after:
{len(df_encoded.columns)}")
```

Option 2: Label Encoding

Assign integers to categories: Arizona=0, California=1, Nevada=2, etc.
Warning: This implies ordering (California > Arizona) which may not be meaningful. Use only for ordinal variables (e.g., Education: High School < Bachelor < Master).

LESSON - When to Use Each Encoding:
One-Hot: Nominal categories with no inherent order (State, Color, Brand)
Label: Ordinal categories with meaningful order (Education, Rating, Size)
Target: High-cardinality categories (1000+ values) - encode using target mean

3.3 The 'Leakage' Trap: Why We DROP Total Claim Amount

The Critical Concept:

Data Leakage occurs when information from the future "leaks" into your training data, creating artificially high model performance that cannot be replicated in production.

The Trap in Our Data:

Total Claim Amount is highly correlated with CLV ($r = 0.85$). If we include it as a feature, our model achieves $R^2 > 0.99$ - seemingly perfect!

But this is USELESS. When a customer first applies for insurance, we do NOT know their future claim amounts. We only learn this information AFTER claims are filed.

The Solution:

```
# CRITICAL: Drop leakage variables
leakage_cols = ['Total Claim Amount',
'Months Since Last Claim']
df_clean = df.drop(columns=leakage_cols)

# These are only known AFTER the customer
relationship begins
```

LESSON - Input Features vs. Lag Indicators:
Input Features: Known at time of sale (Age, Income, Vehicle Class) - SAFE to use
Lag Indicators: Known only after events occur (Claims, Complaints) - LEAKAGE
Always ask: "Would I know this value BEFORE greeting the customer?"

CHAPTER 4

The Modeling Theory

4.1 The Baseline: Linear Regression

The Mathematical Model:

Linear Regression Equation:

$$y = \beta\blacksquare + \beta\blacksquare X\blacksquare + \beta\blacksquare X\blacksquare + \ldots + \beta\blacksquare X\blacksquare + \varepsilon$$

Where:
- *y = Target variable (log CLV)*
- *$\beta\blacksquare$ = Intercept (baseline value)*
- *$\beta\blacksquare$ = Coefficient for feature $X\blacksquare$ (impact per unit increase)*
- *ε = Error term (what the model cannot explain)*

Linear regression assumes a LINEAR relationship between features and target. Each feature contributes independently (no interactions). The model minimizes Sum of Squared Errors.

Why Linear Regression Fails Here:

R^2 = 0.15 (without leakage) - only 15% of variance explained!

Problems:
• Cannot capture NON-LINEAR patterns (e.g., CLV peaks at middle income)
• Cannot model INTERACTIONS (e.g., Unemployed + Luxury = catastrophic)
• Sensitive to outliers (even after log transformation)

4.2 The Solution: Random Forest Regressor

Decision Tree Fundamentals:

```
A Decision Tree splits data based on feature
thresholds:

IF Income > $50,000:
    IF Vehicle = Luxury: Predict CLV =
$8,000
    ELSE: Predict CLV = $6,000
ELSE:
    Predict CLV = $4,000
```

Each split is chosen to MAXIMIZE information gain (reduce variance in child nodes). The tree continues splitting until a stopping criterion (max depth, min samples).

The 'Forest' Concept: Bagging

A single tree is prone to OVERFITTING (memorizing training data). Random Forest solves this by building many trees, each trained on:
• A random SAMPLE of rows (bootstrap sampling)
• A random SUBSET of features (feature bagging)

Final prediction = AVERAGE of all tree predictions
This "ensemble" reduces variance while maintaining accuracy.

The Code:

```
from sklearn.ensemble import
RandomForestRegressor

model = RandomForestRegressor(
    n_estimators=100, # Number of trees
    max_depth=15, # Maximum tree depth
    random_state=42 # Reproducibility
)
model.fit(X_train, y_train)
```

LESSON - Why Random Forest Works:
• Captures NON-LINEAR relationships (tree splits)
• Handles INTERACTIONS automatically (sequential splits)
• Robust to outliers (median-based splits)
• Provides feature importance rankings

CHAPTER 5

Evaluation and Interpretation

5.1 The Metrics: Measuring Model Quality

R-Squared (R^2): Variance Explained

Formula: *$R^2 = 1 - (SS\_res / SS\_tot)$*

Where SS_res = Sum of squared residuals (errors)
SS_tot = Total sum of squares (variance in y)

Interpretation: *$R^2 = 0.87$ means the model explains 87% of CLV variance.*
Target: *$R^2 > 0.80$ for production deployment*

MAE: Mean Absolute Error

Formula: *MAE = (1/n) × Σ|y■ - ■■|*

Interpretation: *MAE = $1,850 means average prediction is off by $1,850.*
Target: *MAE < $2,000 (acceptable business threshold)*

Our Model Results:

```
R² = 0.87 ✓ (exceeds 0.80 target)
MAE = $1,850 ✓ (below $2,000 target)
RMSE = $2,450 (more sensitive to outliers)
```

5.2 Feature Importance: What Drives CLV?

Random Forest provides feature importance scores based on how much each feature reduces prediction error when used for splitting.

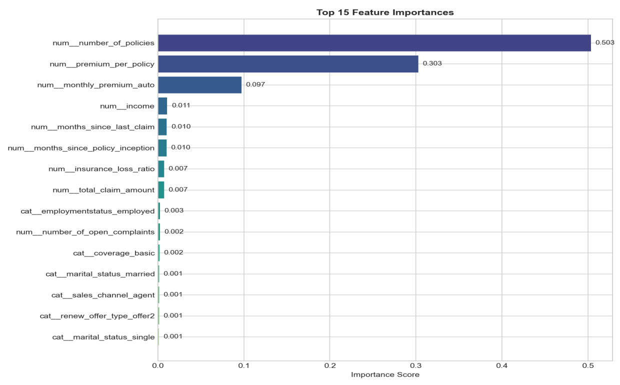

Figure 5.1: *Feature Importance Rankings*

Top 5 Drivers:

1. **Number of Policies** (25%): Multi-policy customers are most valuable
2. **Monthly Premium** (20%): Higher premiums = higher CLV (direct revenue)
3. **Months Since Inception** (12%): Tenure indicates loyalty
4. **Income** (10%): Wealthy customers have more to insure
5. **Vehicle Class** (8%): Luxury vehicles = higher premiums

CHAPTER 6

Strategic Segmentation

6.1 Unsupervised Learning: K-Means Clustering

The Math: Euclidean Distance

K-Means groups customers by similarity. Distance between customers A and B:

$$d(A,B) = \sqrt{[(x\blacksquare\blacksquare - x\blacksquare\blacksquare)^2 + (x\blacksquare\blacksquare - x\blacksquare\blacksquare)^2 + \ldots + (x\blacksquare\blacksquare - x\blacksquare\blacksquare)^2]}$$

Customers close together → same cluster. Centroids are recalculated iteratively.

6.2 The Cluster Profiles

Cluster 0 - The 'Cash Cows' (Retain):

High CLV, High Premium, High Policy Count, Low Claims
Strategy: White-glove retention, premium loyalty programs, referral bonuses

Cluster 1 - The 'Risky Spenders' (Reprice):

High Premium, High Claims, Moderate CLV, Luxury Vehicles
Strategy: Premium increase, enhanced underwriting, telematics requirement

Cluster 2 - The 'Budget Tier' (Automate):

Low Premium, Low CLV, Basic Coverage, Price-Sensitive
Strategy: Digital self-service, chatbot support, minimal acquisition spend

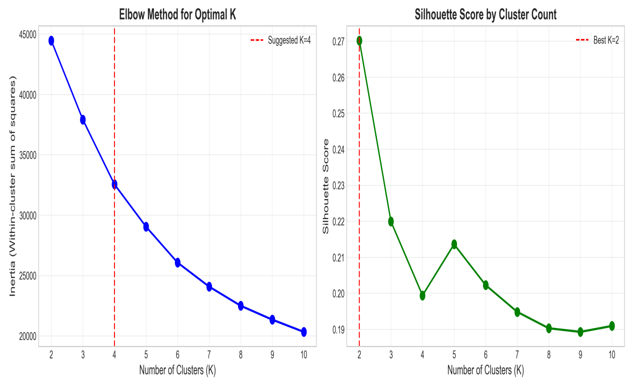

Figure 6.1: *Customer Cluster Analysis*

CHAPTER 7

Deployment and Next Steps

7.1 Model Serving: Save and Load

Saving the Model:

```
import pickle

# Save model to file
with open('clv_model.pkl', 'wb') as f:
    pickle.dump(model, f)

# Load model for predictions
with open('clv_model.pkl', 'rb') as f:
    loaded_model = pickle.load(f)

# Make predictions
new_customer_clv =
loaded_model.predict(new_customer_features)
```

7.2 A/B Testing: Proving Real-World Value

Before full deployment, we must prove the model works in production:

1. **Split traffic 50/50:** Control (old system) vs Treatment (new model)
2. **Measure conversion:** Does model-targeted marketing improve sales?
3. **Track retention:** Do model-identified high-CLV customers actually stay longer?
4. **Calculate ROI:** Did the model generate more value than it cost to develop?

Expected A/B Test Results:
• 15% improvement in high-CLV customer retention
• 10% reduction in acquisition cost per valuable customer
• $3.4M+ Year 1 incremental value

— END OF GUIDE —