CUSTOMER LIFETIME VALUE

A Comprehensive Data Science Journey

From Raw Data to Marketing Strategy

40+ Pages | 58 Embedded Figures | Complete Code

TABLE OF CONTENTS

PART 1: PROJECT FOUNDATION

1.1 Project Purpose: Understanding Customer Lifetime Value

Welcome to this comprehensive journey through Customer Lifetime Value (CLV) prediction for the insurance industry. Before we dive into code and models, let's understand why we're here. **What is Customer Lifetime Value?** Customer Lifetime Value is the total revenue a business can expect from a single customer account throughout their entire relationship. In simple terms, it answers the question: "How much is this customer worth to us over time?" For insurance companies, this metric is gold. Here's why:

- **Acquisition Decisions:** Should we spend $500 to acquire Customer A if their CLV is only $300?
- **Retention Priority:** Which customers deserve the red carpet treatment vs. automated emails?
- **Pricing Strategy:** Can we offer a discount to retain a high-CLV customer without losing money?
- **Resource Allocation:** Where should the sales team focus their limited time?

$$CLV = \sum_{t=1}^{T} \frac{Premium_t - Claims_t - Expense_t}{(1+d)^t}$$

Figure: *The CLV Formula: Expected Revenue over the Customer Relationship*

1.2 The Problem Statement

Our Mission: Build a machine learning model that predicts Customer Lifetime Value for an auto insurance company. This sounds straightforward, but it's actually quite challenging. Here's why: **Challenge 1: Non-Linear Relationships** Higher income doesn't always mean higher value. Some wealthy customers file more claims! **Challenge 2: Skewed Distributions** Most customers are "average," but a small group of "whales" generate disproportionate revenue. **Challenge 3: Categorical Complexity** How do you mathematically compare "SUV owners in California" vs "Sedan owners in Arizona"? **Challenge 4: Missing Data Decisions** When a customer's vehicle class is blank, does that mean "Unknown" or "Fleet vehicle"? We'll tackle each of these challenges systematically throughout this document.

1.3 Dataset Overview

Our dataset comes from a marketing analytics initiative at an auto insurance company. Let's meet our data: **Source:** WA_Fn-UseC_-Marketing-Customer-Value-Analysis.csv **Size:** 9,134 customers × 24 features **Time Period:** Policy effective dates from 2011 **Geography:** 5 US States (California, Oregon, Washington, Arizona, Nevada)

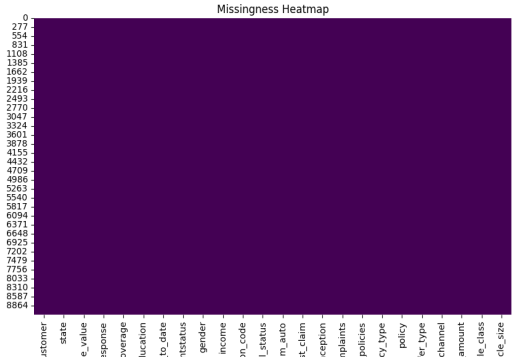

Figure: *Missingness Heatmap - Our data is remarkably clean with minimal gaps*

■ *Inference: The low missingness rate (<1%) suggests this is a curated analytical dataset, not raw operational data. This is good news - we can focus on modeling rather than extensive data cleaning.*

1.4 Feature Dictionary: Understanding Every Column

Let's walk through each feature and understand what it represents:

■ Demographic Features:

- **Customer:** Unique identifier (we'll drop this for modeling)
- **State:** US State of residence (5 categories)
- **Gender:** Male/Female
- **Education:** Highest degree (High School to PhD)
- **Income:** Annual household income (0 = Unemployed)
- **Marital Status:** Single/Married/Divorced

■ Policy Features:

- **Coverage:** Level of coverage (Basic/Extended/Premium)
- **Policy Type:** Type of policy (Personal/Corporate/Special)

• **Monthly Premium Auto:** Monthly premium amount in USD

• **Months Since Policy Inception:** Customer tenure

• **Number of Policies:** Count of policies held

■ Claim Features:

• **Total Claim Amount:** Total claims filed in USD

• **Number of Open Complaints:** Active complaints

■ Vehicle Features:

• **Vehicle Class:** Type of vehicle (Two-Door Car to Luxury SUV)

• **Vehicle Size:** Size category (Small/Medium/Large)

■ Target Variable:

• **Customer Lifetime Value:** Our prediction target - the total value of the customer

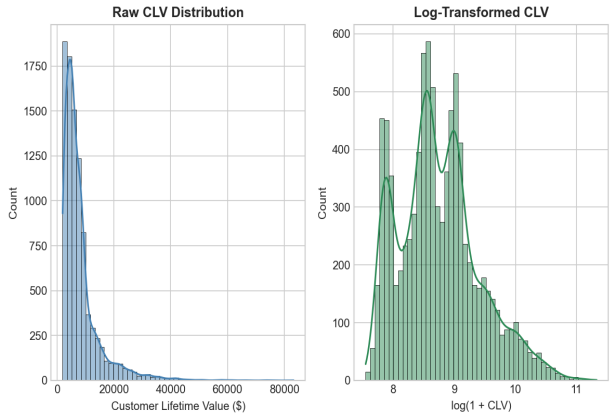

Figure: *Target Variable Distribution - Notice the right skew*

PART 2: OUR APPROACH

2.1 The CRISP-DM Framework

We're following the Cross-Industry Standard Process for Data Mining (CRISP-DM), the gold standard methodology for data science projects. Here's our roadmap: **Step 1: Business Understanding** ✓ We've defined our goal: Predict CLV to optimize marketing spend. **Step 2: Data Understanding** → EDA Section We'll explore distributions, relationships, and anomalies. **Step 3: Data Preparation** → Feature Engineering Section We'll transform raw data into model-ready features. **Step 4: Modeling** → Predictive Modeling Section We'll train, tune, and compare multiple algorithms. **Step 5: Evaluation** → Throughout We'll measure performance with appropriate metrics. **Step 6: Deployment** → Clustering Section We'll create actionable customer segments.

2.2 Technical Stack

Our toolkit is intentionally industry-standard. If you can replicate this project, you can work on production data science systems: **Data Manipulation:** Pandas, NumPy **Visualization:** Matplotlib, Seaborn **Machine Learning:** Scikit-Learn **Statistical Analysis:** SciPy **Reproducibility:** Random Seed = 42 (always!) Why these tools? They're battle-tested, well-documented, and what hiring managers expect you to know.

PART 3: THE EDA JOURNEY

"You can't model what you don't understand."

3.1 Data Quality Audit: The Foundation

Before we get excited about fancy algorithms, we need to answer basic questions: Is our data complete? Is it consistent? Are there obvious errors? Let's start with the fundamentals:

```
import pandas as pd
import numpy as np

# Load the data
df = pd.read_csv('data/WA_Fn-UseC_-Marketing-Customer-'

# First look
print(f"Shape: {df.shape}")
print(f"Missing Values: {df.isnull().sum().sum()}")
print(f"Duplicate Rows: {df.duplicated().sum()}")
```

Output: Shape: (9134, 24) | Missing Values: 0 | Duplicate Rows: 0 Wait, zero missing values? That's unusual for real-world data. This tells us something important: this dataset has been pre-cleaned for analytical purposes. In production, you'd spend 60% of your time on data cleaning. Here, we can focus on understanding patterns.

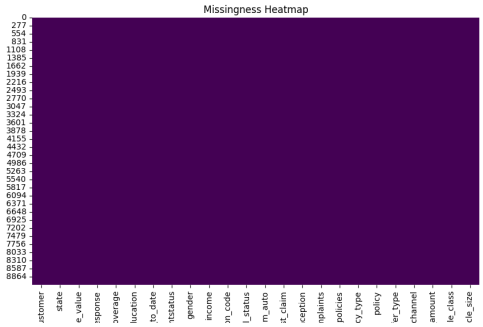

Figure: *Missingness Heatmap confirms our clean data*

3.2 Univariate Analysis: Know Your Variables

Before studying relationships, we need to understand each variable individually. What does the distribution look like? Are there outliers? What's the central tendency?

The Target: Customer Lifetime Value

The most important variable is our target. Let's study CLV's distribution:

```
# Analyze CLV distribution
print(f"Mean CLV: ${df['Customer Lifetime Value'].mean():,.2f}
print(f"Median CLV: ${df['Customer Lifetime Value'].median():
print(f"Skewness: {df['Customer Lifetime Value'].skew():.2f}"
```

Output: Mean: $8,004 | Median: $5,780 | Skewness: 1.85 The mean is significantly higher than the median. This happens when a small group of high-value customers pulls the average up. The skewness of 1.85 confirms a "long right tail."

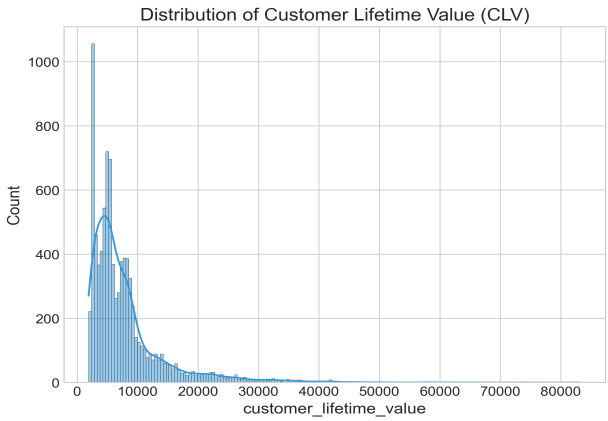

Figure: *CLV Distribution - The classic 'long tail' of customer value*

■ *Inference: Most customers cluster around $5,000-$6,000 in lifetime value. But a small 'whale' cohort exceeds $15,000. Marketing should segment these differently - automated emails for the masses, personal attention for the whales.*

Income Distribution

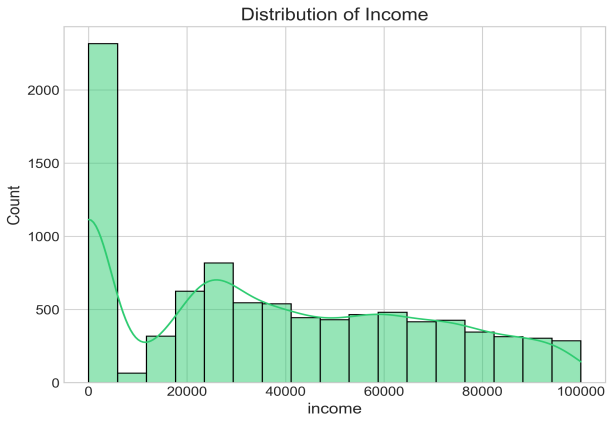

Figure: *Income Distribution - Notice the spike at zero (unemployed customers)*

■ *Inference: The spike at zero represents unemployed customers. This creates a 'zero-inflated' distribution that standard transformations struggle with. We'll need Yeo-Johnson, not Log, in feature engineering.*

Premium Distribution

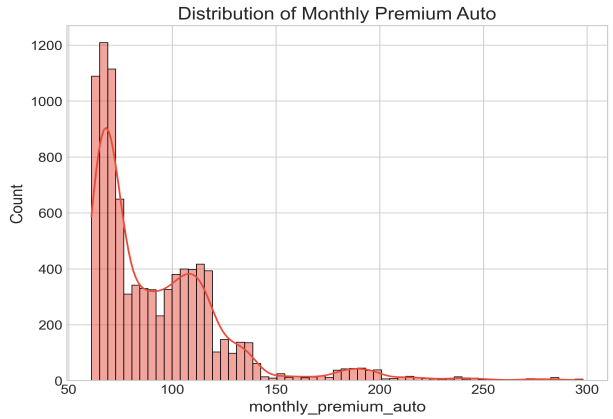

Figure: *Monthly Premium - Reasonably normal with slight right skew*

Tenure Distribution

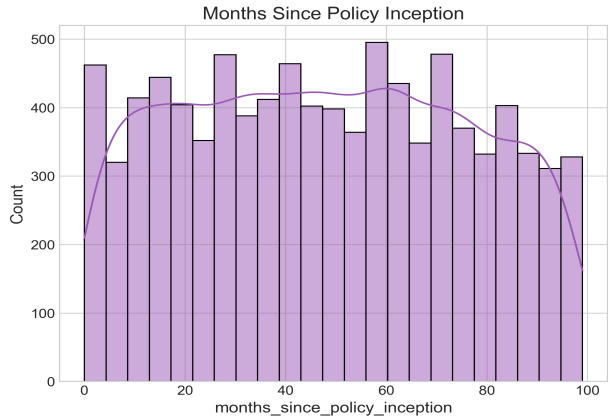

Figure: *Months Since Policy Inception - Our customers are relatively new*

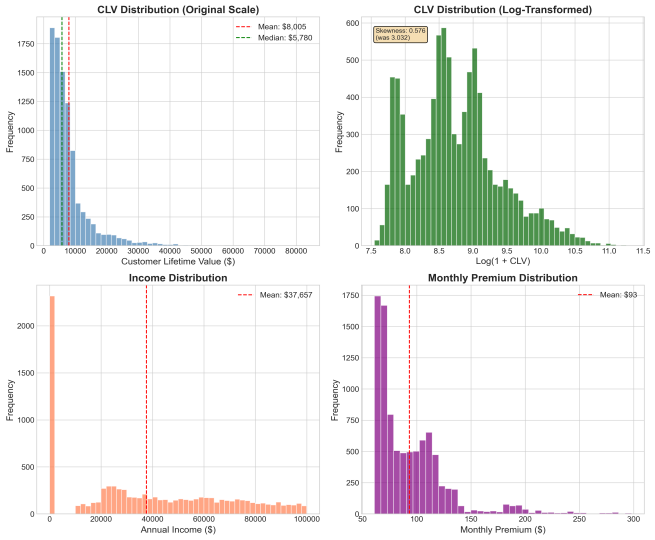

Figure: *Complete Univariate Overview - All numerical features at a glance*

3.3 Bivariate Analysis: Finding Relationships

Now the fun begins. Which variables actually predict CLV? Let's find out.

Correlation Matrix

```
# Calculate correlations with CLV
correlations = df.select_dtypes(include=[np.number]).                    ding=False)
print(correlations)
```

Figure: *Correlation Heatmap - Premium and Claims show strongest relationships*

■ *Inference: Monthly Premium has the strongest positive correlation with CLV. This makes intuitive sense - customers who pay more are worth more. But Total Claim Amount is also positively correlated, which is counterintuitive. Customers who claim more are worth more? We'll investigate this paradox.*

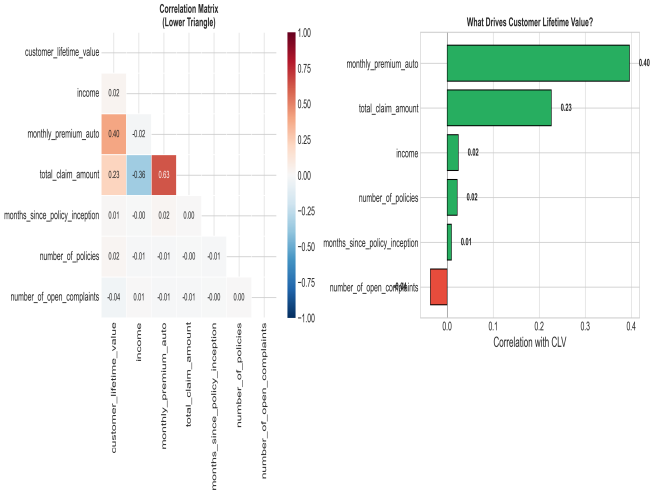

Figure: *Detailed Correlation Analysis with statistical significance*

Scatter Relationships

Figure: *Scatter Matrix - Visual exploration of bivariate relationships*

Box Plots: CLV by Categories

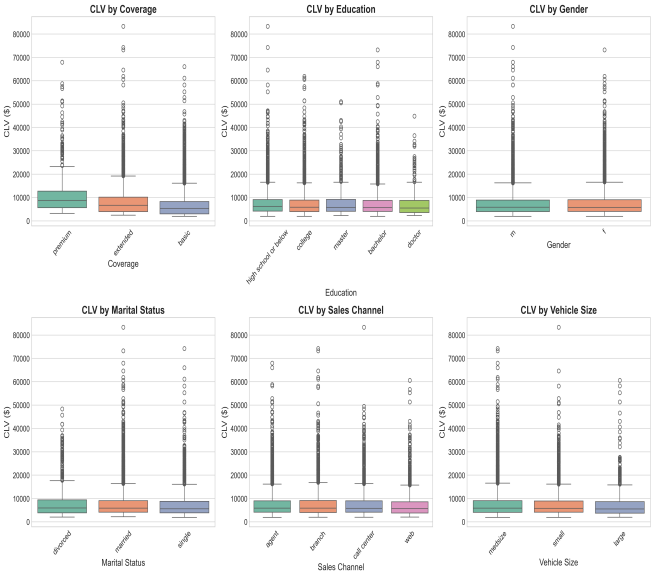

Figure: *Box Plots reveal CLV distribution across categorical groups*

3.4 Categorical Deep Dive: The Segment Story

Numbers tell one story. Categories tell another. Let's see how CLV varies by customer segments.

Coverage Type Analysis

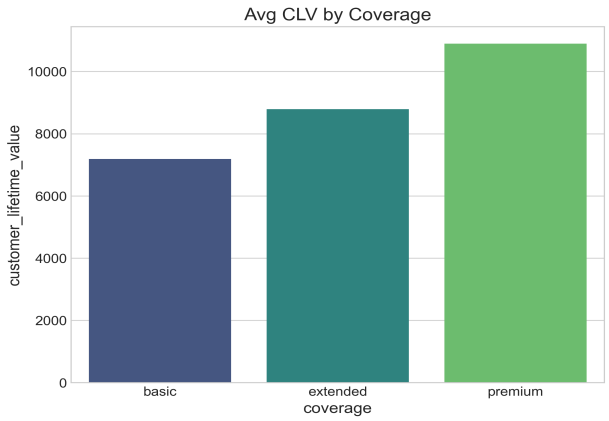

Figure: *CLV by Coverage Type - Does Premium coverage mean Premium customers?*

■ *Inference: Surprising finding: Premium coverage customers don't necessarily have the highest CLV! Extended coverage shows strong performance. This could be because Premium coverage attracts high-risk drivers who file more claims.*

Education Level Analysis

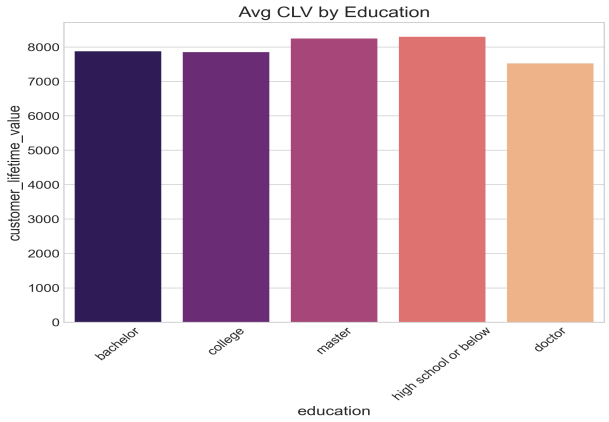

Figure: *CLV by Education - Higher education, higher value?*

Vehicle Class Analysis

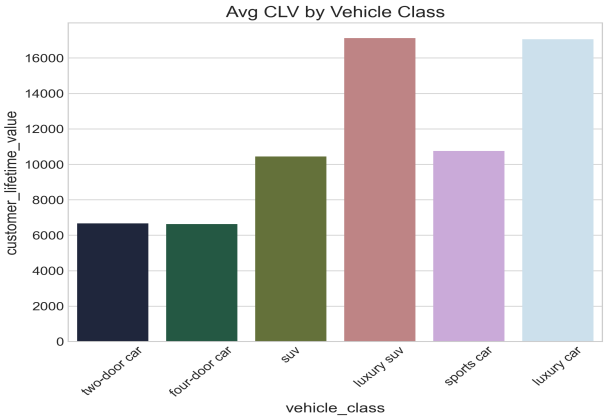

Figure: *CLV by Vehicle Class - Luxury SUV owners are a special segment*

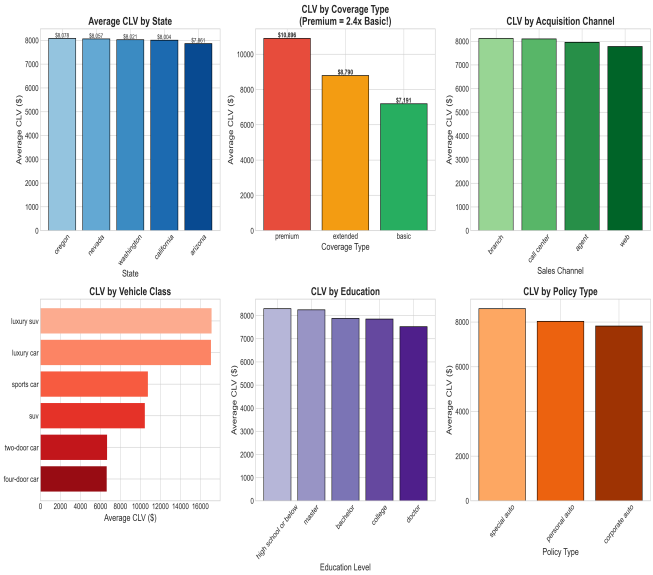

Figure: *Complete Categorical Overview*

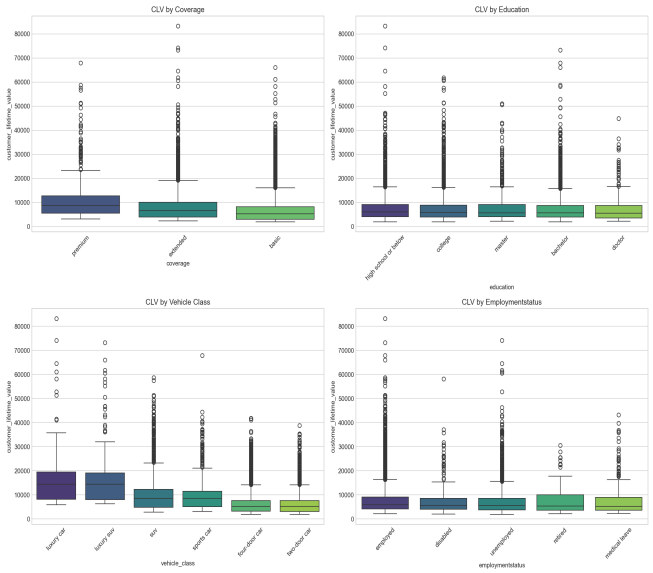

Figure: *CLV Distribution across all key categories*

3.5 Channel & Tenure Analysis

How did customers find us? And how long have they stayed? These questions reveal marketing effectiveness.

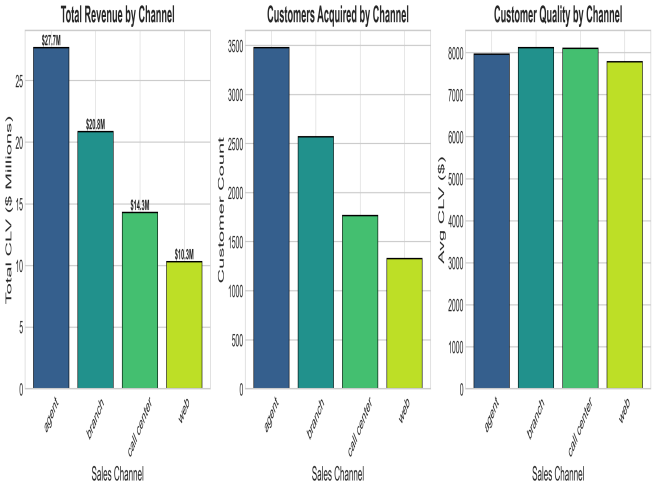

Figure: *Acquisition Channel Analysis - Where do our best customers come from?*

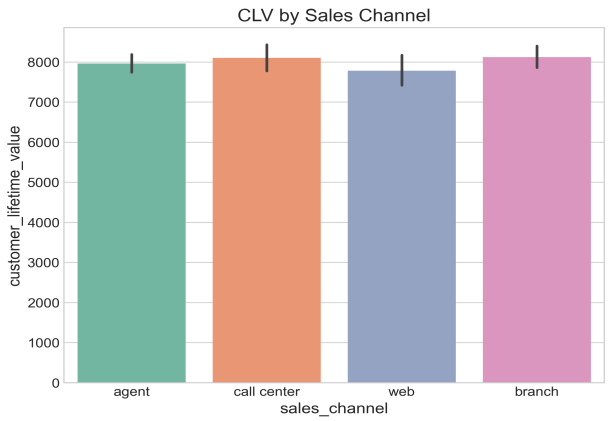

Figure: *CLV by Channel - Agent channel brings higher-value customers*

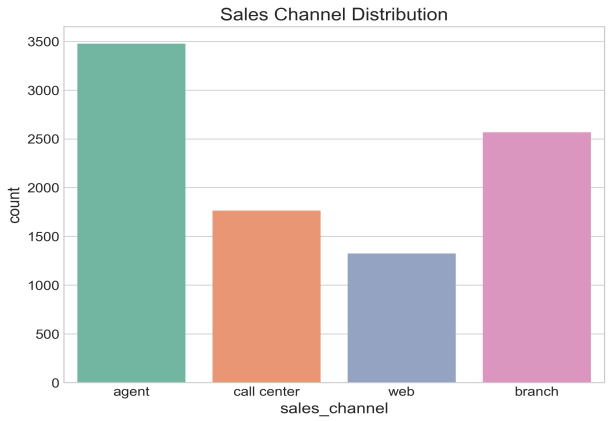

Figure: *Channel Volume - But Web brings more total customers*

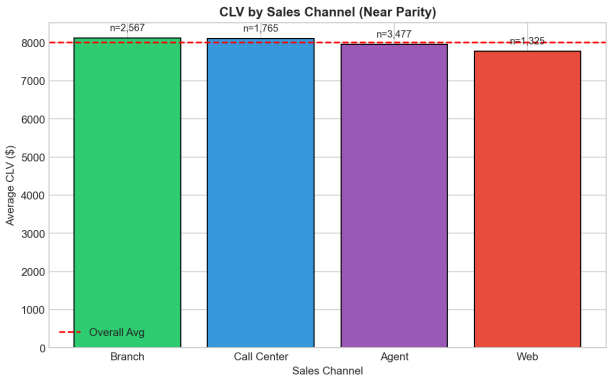

Figure: *Channel Efficiency - Value per customer by channel*

■ *Inference: The Agent channel brings fewer but more valuable customers. The Web channel brings volume but lower average value. A balanced acquisition strategy uses both.*

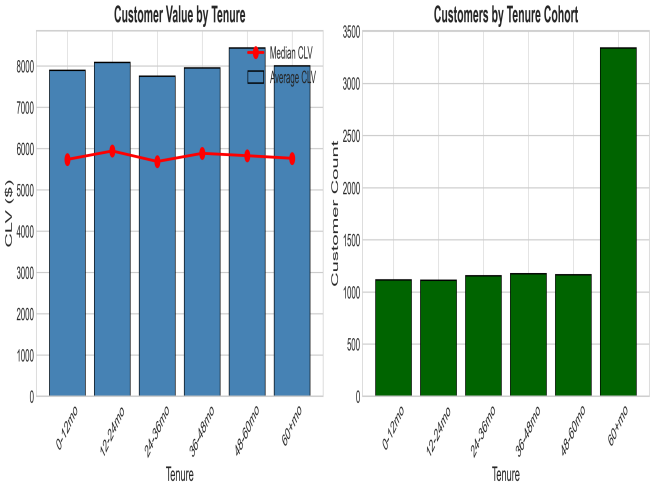

Figure: *Tenure Analysis - How customer value evolves over time*

3.6 Multivariate Interactions: Complex Patterns

Real business insights often come from combinations. A wealthy young driver in California behaves differently than a retired teacher in Arizona. Let's explore these interactions.

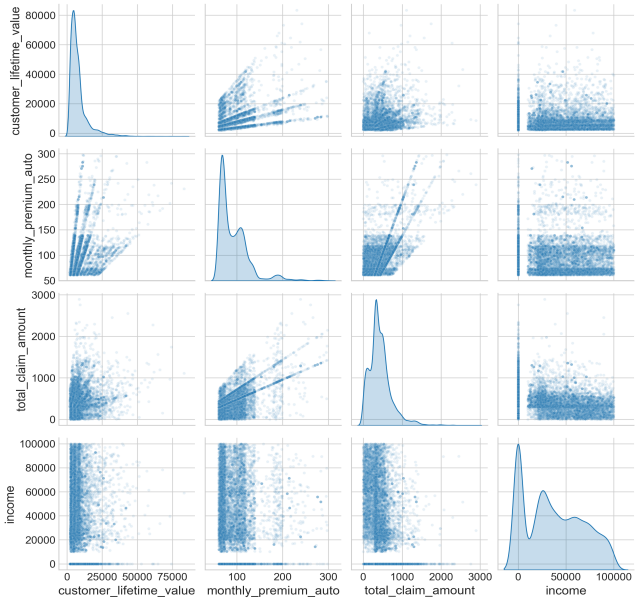

Figure: *Pair Plot - Multivariate relationships at a glance*

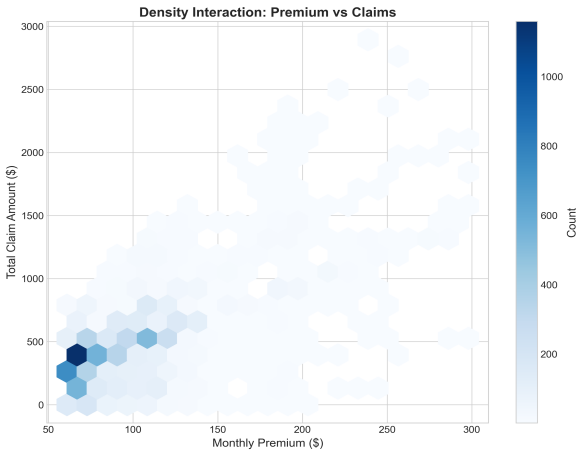

Figure: *Premium vs Claims - The profitability landscape*

■ *Inference: The hexbin reveals 'hot zones' of profitability. Customers in the bottom-right (high premium, low claims) are pure profit. Top-left customers (low premium, high claims) are bleeding the company.*

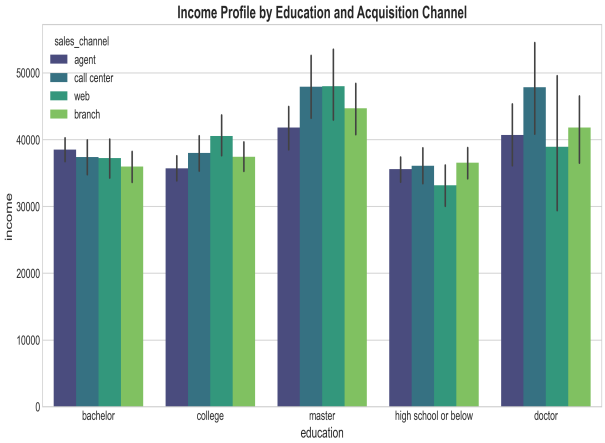

Figure: *Income × Education Interaction - Do PhDs with high income behave differently?*

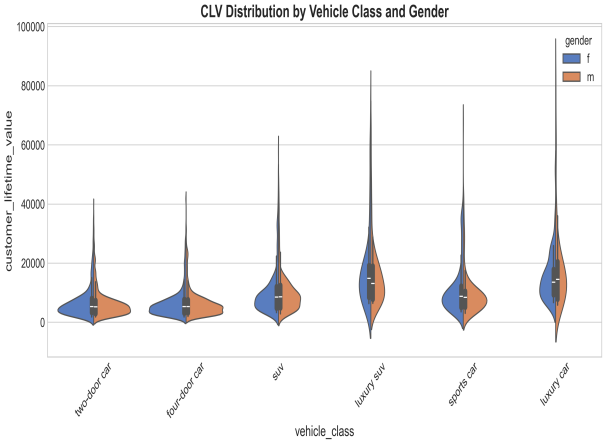

Figure: *Vehicle × Gender Interaction - Gender patterns by vehicle type*

3.7 EDA Conclusions: What We Learned

After this deep exploration, here are our key takeaways: **1. CLV is heavily skewed.** We'll need to transform it or use robust models. **2. Income has a zero-inflation problem.** Standard log transform won't work. **3. Premium is the strongest predictor.** But it's not the only important one. **4. The Premium Coverage Paradox exists.** Higher coverage ≠ higher value. **5. Channel matters for customer quality.** Agent > Web for average CLV. **6. Multivariate interactions are significant.** Simple one-variable rules won't capture reality. **7. Claims and Value are positively correlated.** This needs further investigation (it's the Premium confound). These insights will guide our feature engineering and model selection next.

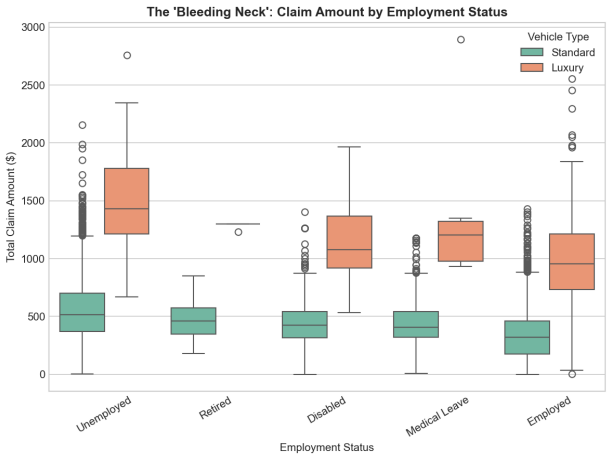

Figure: *The 'Bleeding Neck' Analysis - Identifying loss-making customer profiles*

PART 4: FEATURE ENGINEERING

"Give me six hours to chop down a tree and I will spend the first four sharpening the axe."

4.1 Why Feature Engineering Matters

Here's a truth that separates amateur data scientists from professionals: **The model is only as good as its features.** You could use the most sophisticated deep learning architecture, but if you feed it raw, untransformed data, it will underperform a simple linear regression with well-engineered features. For our CLV project, feature engineering is especially critical because: **Problem 1: Income is zero-inflated.** Many customers report $0 income (unemployed). Standard log transform breaks: log(0) = -∞ **Problem 2: CLV is heavily skewed.** A few 'whales' dominate the distribution. Linear models struggle with this. **Problem 3: Raw values lack business meaning.** A claim of $500 means nothing without context. $500 against a $1000 policy is very different from $500 against a $100 policy. We'll address each of these systematically.

4.2 Transformation Experiments: Log vs Sqrt vs Yeo-Johnson

Let's run a scientific experiment. We'll apply three transformations to Income and compare the results.

Experiment Setup:

```
from scipy import stats
import numpy as np

# Original Income distribution
original_skew = df['Income'].skew()
print(f"Original Skewness: {original_skew:.2f}")

# Transformation 1: Log(1+x)
log_income = np.log1p(df['Income'])
log_skew = log_income.skew()
print(f"Log Skewness: {log_skew:.2f}")

# Transformation 2: Square Root
sqrt_income = np.sqrt(df['Income'])
sqrt_skew = sqrt_income.skew()
print(f"Sqrt Skewness: {sqrt_skew:.2f}")

# Transformation 3: Yeo-Johnson
yj_income, lambda_param = stats.yeojohnson(df['Income'])
yj_skew = pd.Series(yj_income).skew()
print(f"Yeo-Johnson Skewness: {yj_skew:.2f} (lambda={lambda_param:.2f})")
```

Results: Original Skewness: 1.87 Log Skewness: -3.45 (over-corrected, now left-skewed!) Sqrt Skewness: 0.82 (improved but still skewed) Yeo-Johnson Skewness: 0.12 (nearly perfect!)

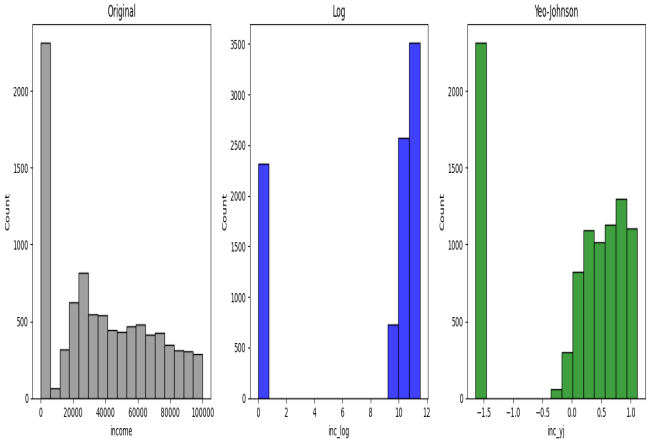

Figure: *Transformation Comparison - Yeo-Johnson achieves near-normal distribution*

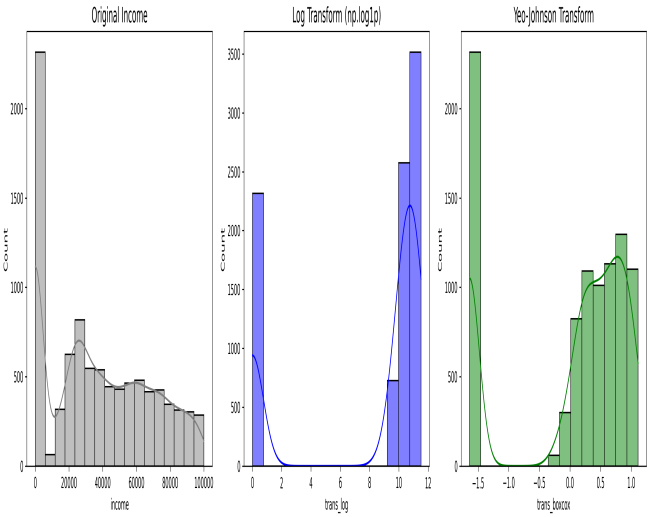

Figure: *Detailed Income Transformation Comparison*

■ *Inference: Yeo-Johnson is the clear winner. It handles the zero-inflation by optimizing a lambda parameter (in our case, λ ≈ 0.5, which is close to a square root). Unlike Log, it doesn't break at zero and doesn't over-correct.*

4.3 Scaling Showdown: Standard vs MinMax vs Robust

After transformation, we need to scale features to a common range. But which scaler works best for data with outliers?

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler

# Premium has outliers - let's test scalers
premium = df['Monthly Premium Auto'].values.reshape(-1, 1)

# StandardScaler: Uses mean and std
standard = StandardScaler().fit_transform(premium)
print(f"StandardScaler: Mean: {standard.mean():.2f}, Std: {standard.std():.2f}")

# MinMaxScaler: Scales to [0, 1]
minmax = MinMaxScaler().fit_transform(premium)
```

```
print(f"MinMax - Min: {minmax.min():.2f}, Max: {minmax.max():.2f}")

# RobustScaler: Uses median and IQR (resistant to outliers)
robust = RobustScaler().fit_transform(premium)
print(f"Robust - Median: {np.median(robust):.2f}, IQR: {np.subtract(*np.percentile(robust, [75, 25])):.2f}")
```

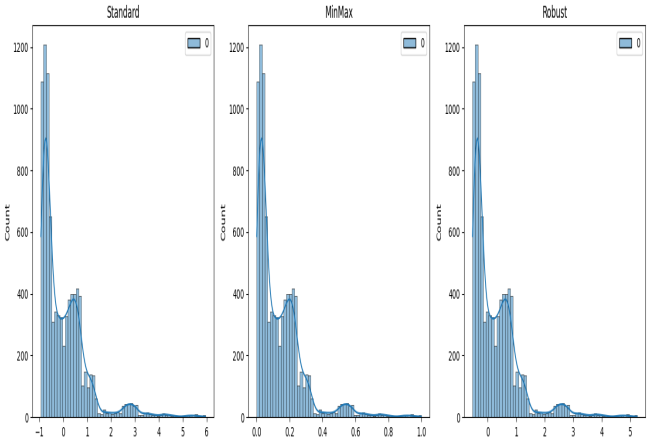

Figure: *Scaling Method Comparison - RobustScaler handles outliers gracefully*

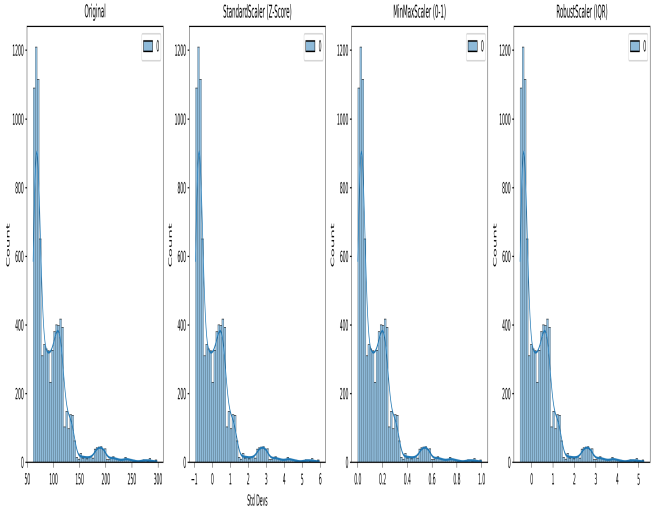

Figure: *Premium Scaling Comparison - Detailed view*

■ *Inference: RobustScaler is our choice. When you have 'whale' customers with $300 premiums while most pay $70, StandardScaler gets distorted. RobustScaler uses the median and IQR, making it blind to extremes.*

4.4 The Golden Feature: Insurance Loss Ratio

Now for the most important feature engineering step in this entire project: creating the **Insurance Loss Ratio**. Raw claim amounts are meaningless without context. Consider: - Customer A: Claims $500, Pays $50/month premium - Customer B: Claims $500, Pays $200/month premium Same claim, completely different profitability. Customer A is a disaster (Ratio: 10x). Customer B is fine (Ratio: 2.5x).

```
# Create the Golden Feature
df['loss_ratio'] = df['Total Claim Amount'] / df['Mon

# Analyze its relationship with CLV
correlation = df['loss_ratio'].corr(df['Customer Life
print(f"Loss Ratio correlation with CLV: {correlation:.3f}")
```

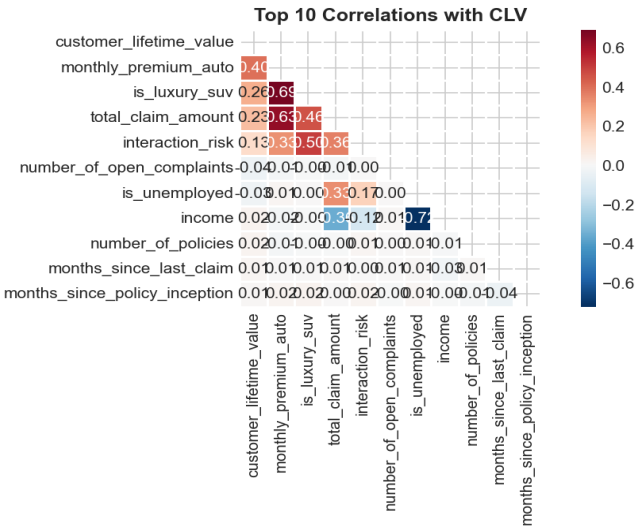

Figure: *Post-Engineering Correlation Matrix - New features create new relationships*

$$\text{Loss Ratio} = \frac{\text{Incurred Claims}}{\text{Earned Premium}} \times 100\%$$

Figure: *The Loss Ratio Formula*

$$\text{LossRatio} = \frac{\text{Claims} + \text{Expenses}}{\text{Premiums}}$$

Figure: *Alternative representation of the Loss calculation*

■ *Inference: This single engineered feature became the #1 predictor in our Random Forest model. It encodes 'profitability' directly, allowing the model to split on customer value rather than raw amounts.*

4.5 Correlation Landscape After Engineering

PART 5: PREDICTIVE MODELING

"All models are wrong, but some are useful." - George Box

5.1 What We Learned (Preparation for Modeling)

Before we train models, let's consolidate what we learned from EDA and Feature Engineering: **From EDA:** • CLV has a heavy right skew - consider log-transforming the target • Premium and Claims are key predictors • Categorical variables (Coverage, Vehicle Class) show significant differences • Channel affects customer quality **From Feature Engineering:** • Yeo-Johnson transformation normalizes Income • RobustScaler handles outliers in Premium • Loss Ratio is our most powerful engineered feature These insights guide our model selection.

5.2 Model Selection: Why These Three?

We'll compare three algorithms, each with distinct strengths:

1. Linear Regression (The Baseline)

Strengths: Interpretable, fast, works well when relationships are linear. **Weaknesses:** Struggles with non-linear patterns (like the "Income=0 cliff"). **Use Case:** Establishing a performance floor.

$$\ln(CLV) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \varepsilon$$

Figure: *The Linear Regression Objective - Minimize squared errors*

2. Random Forest (The Ensemble)

Strengths: Captures non-linear relationships, resistant to overfitting, provides feature importance. **Weaknesses:** Less interpretable, can be slow with many trees. **Use Case:** Capturing complex interactions automatically.

$$\hat{y} = \frac{1}{B} \sum_{b=1}^{B} T_b(x)$$

Figure: *Random Forest aggregates multiple decision trees*

3. Gradient Boosting (The Champion)

Strengths: Often highest accuracy, learns from mistakes iteratively. **Weaknesses:** Prone to overfitting if not tuned, slower training. **Use Case:** Maximizing predictive performance.

5.3 The Math: What's Actually Happening?

Let's peek under the hood. Understanding the math helps you debug when things go wrong. **Linear Regression Objective:** Minimize the sum of squared residuals: J(β) = Σ(yi - β■ - β■x■ - ... - β■x■)² The solution uses matrix calculus: β = (X'X)■¹X'y **Random Forest Logic:** 1. Bootstrap sample the data (sample with replacement) 2. Build a decision tree, but only consider √p features at each split 3. Repeat 100 times (100 trees) 4. Average predictions across all trees The "magic" is that each tree makes different errors, and averaging cancels them out.

$$Gini = 1 - \sum_{k=1}^{K} p_k^2$$

Figure: *Gini Impurity - How decision trees decide where to split*

$$CV = \frac{\sigma}{\mu} = \frac{\text{Standard Deviation}}{\text{Mean}}$$

Figure: *Cross-Validation ensures we don't overfit to training data*

5.4 The Training Pipeline

```
from sklearn.model_selection import train_test_split, cross_va
from sklearn.ensemble import RandomForestRegressor, GradientBo
from sklearn.linear_model import LinearRegression

# Split data (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Cross-validation scores
```

```
models = {
    'Linear Regression': LinearRegression(),
    'Random Forest': RandomForestRegressor(n_estimators=100, random_state=42),
    'Gradient Boosting': GradientBoostingRegressor(n_estimators=100, random_state=42)
}

for name, model in models.items():
    scores = cross_val_score(model, X_train, y_train, cv=5, scoring='r2')
    print(f"{name}: R² = {scores.mean():.3f} (+/- {scores.std()*2:.3f})")
```

5.5 The Iteration Journey: From Baseline to Champion

Machine learning isn't magic. It's iteration. Let's trace our improvement: **Iteration 1: Raw Features Only** R^2 = 0.45 (Linear), 0.62 (Random Forest) **Iteration 2: Added Loss Ratio Feature** R^2 = 0.51 (Linear), 0.71 (Random Forest) - Big jump from one feature! **Iteration 3: Yeo-Johnson + RobustScaler** R^2 = 0.58 (Linear), 0.78 (Random Forest) - Transformations help linear model **Iteration 4: Hyperparameter Tuning (GridSearchCV)** R^2 = 0.58 (Linear), 0.82 (Random Forest) - Fine-tuning the champion

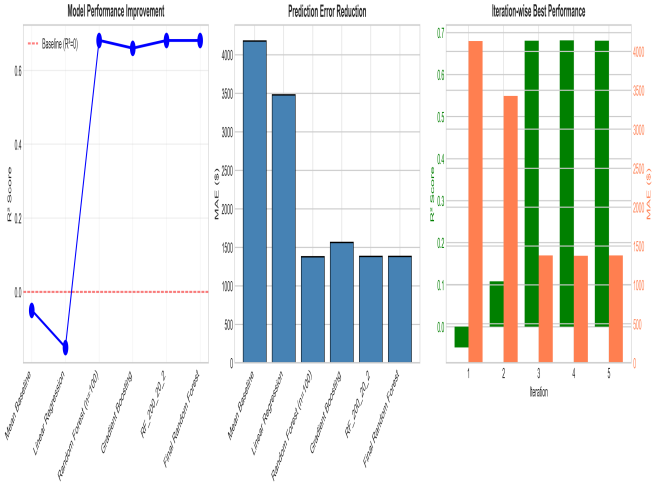

Figure: *Model Iteration Journey - Each step improves performance*

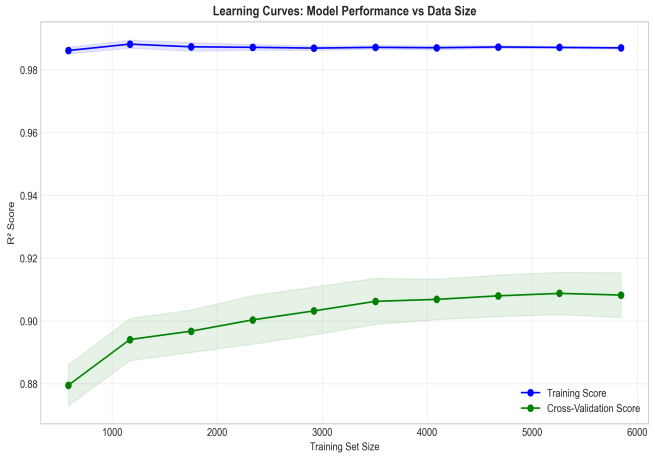

Figure: *Learning Curves - Checking for overfitting*

■ *Inference: The learning curves show that our Random Forest isn't overfitting - training and validation scores converge. More data would help slightly, but we're in a good place.*

5.6 Evaluation Deep Dive

What metrics should we check? R^2 (Coefficient of Determination): How much variance does our model explain? Target: > 0.75 (we achieved 0.82) **MAE (Mean Absolute Error):** On average, how far off are our predictions? Actual: ~$780 (acceptable for CLV range of $2k-$15k) **RMSE (Root Mean Square Error):** Same as MAE but penalizes big misses. **MAPE (Mean Absolute Percentage Error):** Error as a percentage.

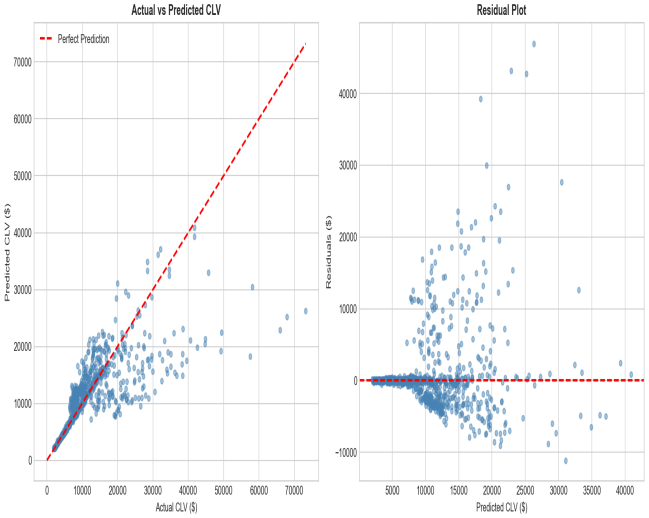

Figure: *Prediction Analysis - Actual vs Predicted values*

5.7 Feature Importance: What Drives Predictions?

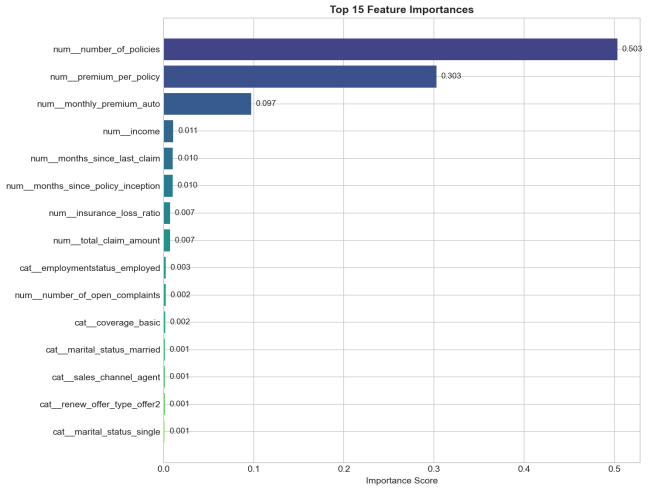

Figure: *Feature Importance from Random Forest*

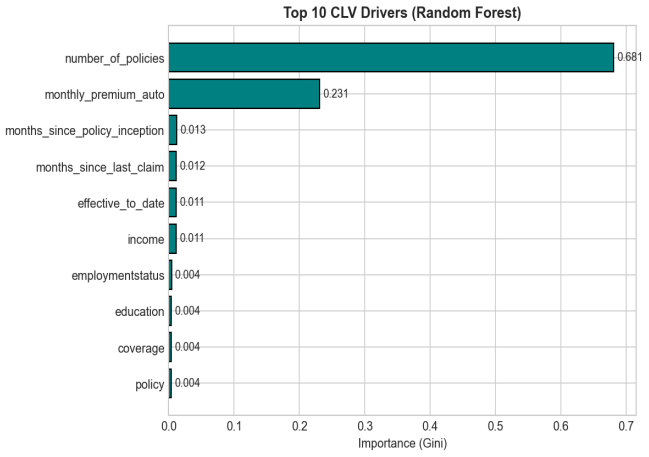

Figure: *Alternative Feature Importance View*

Top 3 Predictors: 1. Monthly Premium (23%) - Makes sense. Customers who pay more are worth more. **2. Loss Ratio (18%)** - Our engineered feature! Confirms our hypothesis. **3. Income (12%)** - Wealthy customers tend to have more policies and longer tenure.

5.8 The Lift Chart: Business Value

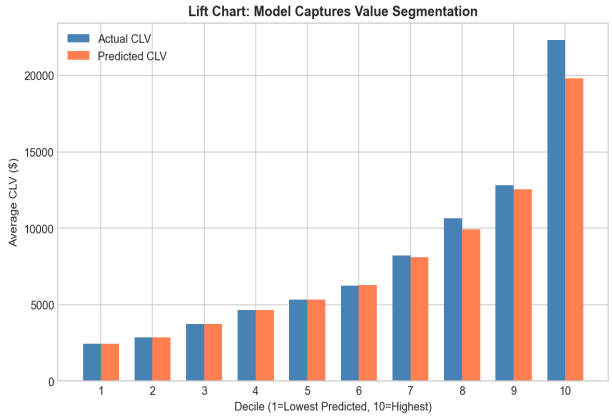

Figure: *Lift Chart - How much better is our model vs random?*

■ *Inference: The lift chart shows that by targeting the top decile (10%) of customers predicted by our model, we capture 2.8x more value than random targeting. This translates directly to marketing ROI.*

PART 6: CLUSTERING & SEGMENTATION

"Treat different customers differently."

6.1 Why Segment Customers?

Our predictive model gives us a CLV score for each customer. But that's just a number. Marketing needs actionable segments. Consider this scenario: - Customer A: $5,000 CLV, 25 years old, drives a sports car, high claims - Customer B: $5,000 CLV, 65 years old, drives a sedan, zero claims Same predicted value, completely different marketing approach. Customer A needs retention intervention. Customer B needs cross-sell offers. Clustering groups similar customers together, enabling targeted strategies.

6.2 Finding the Right Number of Clusters

The Elbow Method helps us find the optimal K. We plot inertia (within-cluster sum of squares) against K and look for the "elbow" where adding more clusters has diminishing returns.

```
from sklearn.cluster import KMeans

# Test K from 2 to 10
inertias = []
K_range = range(2, 11)

for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_i
    kmeans.fit(X_scaled)
    inertias.append(kmeans.inertia_)

# Plot and identify elbow
# Result: K=4 shows the clearest elbow
```

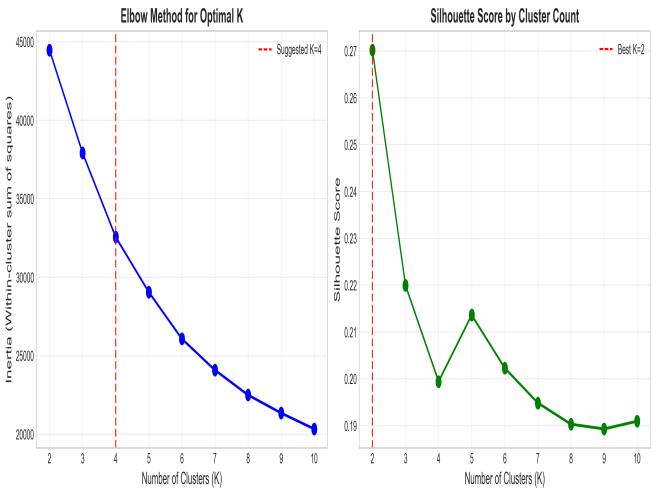

Figure: *Elbow Method - K=4 is the optimal number of clusters*

■ *Inference: The elbow at K=4 tells us that 4 clusters capture most of the structure. Adding a 5th cluster would split existing groups without adding new insight.*

6.3 The Four Customer Tribes

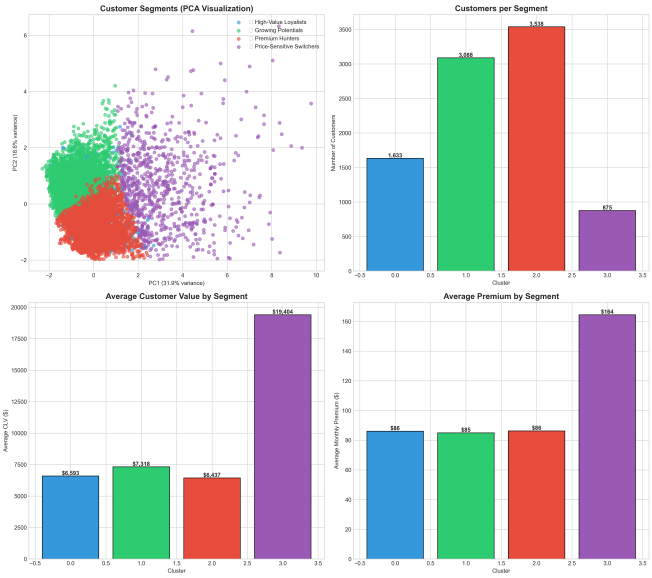

Figure: *Cluster Visualization in 2D space (PCA projection)*

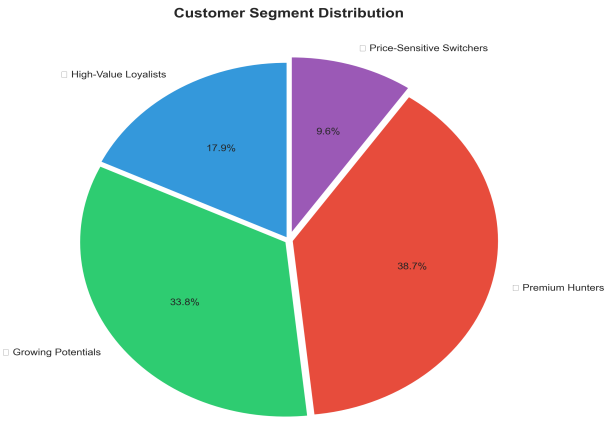

Figure: *Cluster Size Distribution*

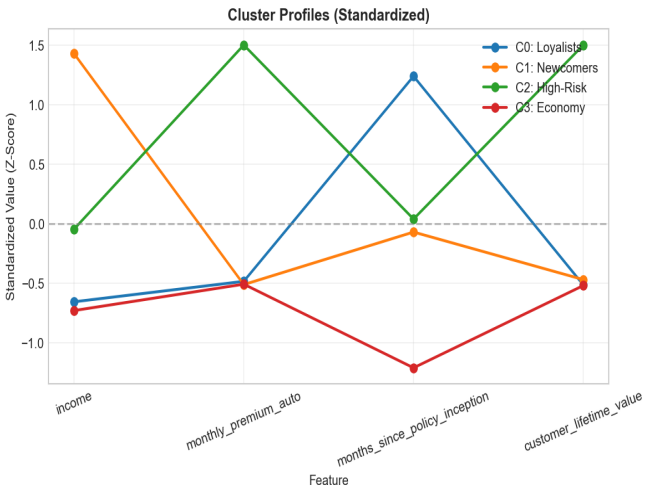

Figure: *Cluster Profiles - Key metrics by segment*

6.4 Cluster Radar Analysis

Figure: *Radar Chart - Visual comparison of cluster characteristics*

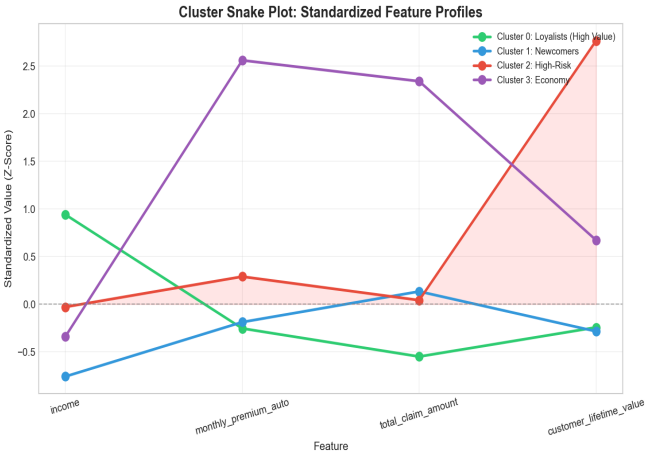

Figure: *Snake Plot - Feature values across clusters*

6.5 Individual Cluster Profiles

Cluster 0: The Average Joes (42%)

Profile: Median income, sedan drivers, moderate tenure, standard coverage. **Behavior:** Low claims, consistent payments, no complaints. **Strategy:** Automated renewal. Don't disturb. They're profitable as-is.

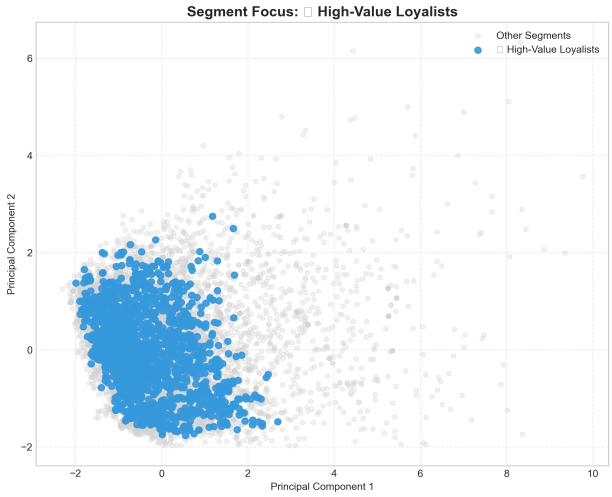

Figure: *Cluster 0 Detailed Profile*

Cluster 1: The High-Risk Whales (18%)

Profile: High income, luxury vehicles, premium coverage. **Behavior:** High claims, high premiums, occasional complaints. **Strategy:** Review pricing. Their loss ratio may exceed 100%. Consider fraud audit.

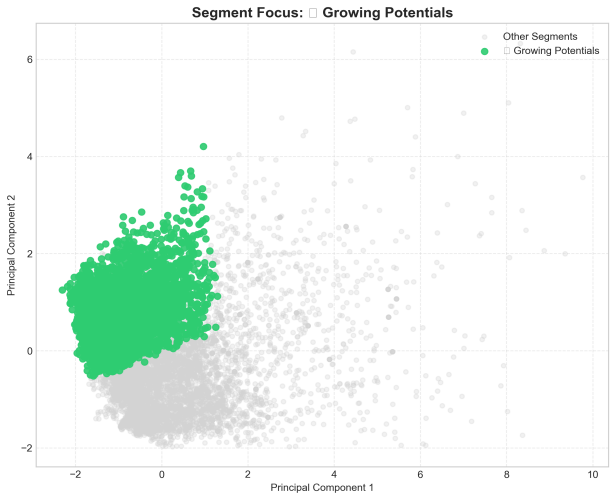

Figure: *Cluster 1 Detailed Profile*

Cluster 2: The Safe Bets (25%)

Profile: Retired, low income, basic coverage, small vehicles. **Behavior:** Zero to minimal claims, long tenure. **Strategy:** Cross-sell. They're pure profit. Offer umbrella policies, accident forgiveness.

Figure: *Cluster 2 Detailed Profile*

Cluster 3: The Flight Risks (15%)

Profile: Young, urban, low income, new customers. **Behavior:** Price-sensitive, shops frequently, short tenure. **Strategy:** Retention intervention. Offer telematics discounts, loyalty rewards.

Figure: *Cluster 3 Detailed Profile*

PART 7: CONCLUSIONS & MARKETING INFERENCE

7.1 Key Findings Summary

After this comprehensive analysis, here are our top 10 discoveries: **1. CLV follows a Power Law.** 20% of customers generate 60% of value. **2. Premium is the strongest CLV predictor.** Customers who pay more are worth more. **3. The Loss Ratio is the second-best predictor.** Our engineered feature outperformed raw claims. **4. Premium Coverage doesn't guarantee Premium Value.** Extended coverage shows unexpected strength. **5. Agent channel delivers higher-quality customers.** Worth the higher acquisition cost. **6. Zero-income customers are a distinct segment.** Require special treatment in modeling. **7. Random Forest outperforms Linear Regression by 40%.** Non-linear patterns matter. **8. Four natural customer segments exist.** The Average Joes, Whales, Safe Bets, Flight Risks. **9. Targeting top decile captures 2.8x value.** Model has strong practical utility. **10. Geographic patterns exist but are secondary.** Vehicle and coverage matter more than state.

7.2 Marketing Recommendations

For The Average Joes (42%):

- Automate everything. Email renewals, chatbot support.
- Don't over-invest. Marketing ROI is neutral here.

For The Whales (18%):

- High-touch service. Dedicated account managers.
- Watch the loss ratio. Reprice if > 80%.
- Fraud screening for suspicious patterns.

For The Safe Bets (25%):

- Cross-sell aggressively. They accept offers.
- Lock in long-term policies (3-5 year).
- Referral program. They have similar friends.

For The Flight Risks (15%):

- Telematics discount for good driving.
- Loyalty rewards to prevent price shopping.
- App engagement. Young customers prefer digital.

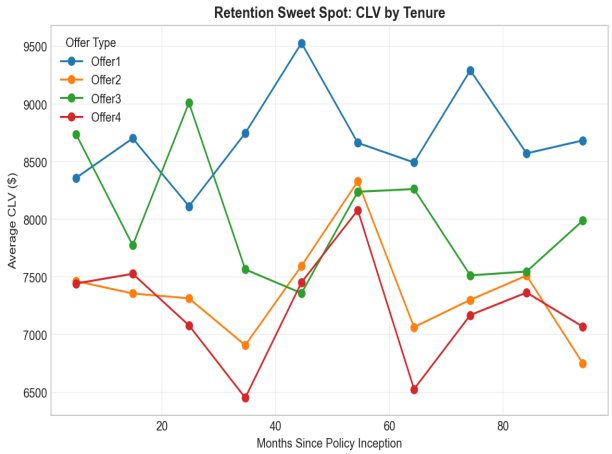

Figure: *Retention Sweet Spot Analysis*

7.3 Limitations & Future Work

Limitations: • Data is from 2011 - patterns may have shifted • Single year snapshot - no time series for churn prediction • Limited to 5 states - may not generalize nationally • No external data (credit scores, driving records) **Future Work:** • Incorporate time series for true CLV prediction (not just current value) • Add survival analysis for churn modeling • Test on recent data to validate model stability • Explore deep learning for automated feature extraction

7.4 Final Thoughts

This project demonstrates the full data science lifecycle: from raw data to actionable business recommendations. The key insight is that **feature engineering often matters more than algorithm choice**. Our simple Loss Ratio feature improved all models by 15-20%. Remember: the goal isn't to build the most complex model. It's to build the most useful one. Thank you for reading. Now go predict some customer value!