THE TECHNICAL MEMOIR

A Forensic Audit of Customer Value

CHAPTER: 01 Data Loading And Cleaning

01. Data Loading & Forensic Audit **Purpose**: Forensic Analysis **Reproducibility**: Random Seed=42

Audit Step 1:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os

# Configuration
SEED = 42
np.random.seed(SEED)
pd.set_option('display.max_columns', None)
```

The Logic: Visualization Step. We generate a graphical representation to perceive patterns that are invisible in tabular format.

1.1 Data Ingestion Goal: Load the raw CSV and establish the baseline.

Audit Step 2:

```
df = pd.read_csv('../WA_Fn-UseC_-Marketing-Customer-Value-Analysis.csv')
```

The Logic: We ingest the 9,134-row dataset. We explicitly use Pandas (Python Data Analysis Library) because its C-optimized backend handles type inference efficiently. **The Output:** A DataFrame object is created in memory, holding the raw "Customer Value" text. **The Impact:** This is the foundation. Any corruption here (encoding errors, delimiters) would poison the downstream model.

1.2 Structure Audit Goal: Determine the dimensionality of the dataset.

Audit Step 3:

```
print(f'Rows: {df.shape[0]}, Columns: {df.shape[1]}')
```

The Logic: Dimensionality Check. We verify (Rows, Columns). **The Output:** (9134, 24). **The Impact:** We have enough samples (N > 5000) to rely on the Central Limit Theorem. We check columns to ensure no features were dropped during ingestion.

1.3 Sample Inspection Goal: Verify data parsing and header alignment.

Audit Step 4:

```
df.head(5)
```

The Logic: Data Manipulation Step. **The Why:** This transformation is required to prepare the feature vector for the subsequent mathematical operation. **The Impact:** Ensures the data structure conforms to the requirements of the Scikit-Learn Estimator API.

1.4 Header Standardization Goal: Normalize column names to snake_case for code consistency.

Audit Step 5:

```
df.columns = [c.lower().replace(' ', '_') for c in df.column
print(df.columns.tolist())
```

The Logic: Data Cleaning. We standardize categorical strings. **The Output:** Uniform labels (e.g., 'California' vs 'california'). **The Impact:** Reduces cardinality and prevents split-feature bugs in One-Hot Encoding.

1.5 Data Type Audit Goal: Identify schema mismatches (e.g. numeric columns stored as objects).

Audit Step 6:

```
df.info()
```

The Logic: Type Audit. We check if numerical columns (like 'Income') were parsed as Strings (Objects). **The Output:** A schema summary. **The Impact:** Mismatched types are the #1 cause of model failure. 'Effective To Date' usually fails here, appearing as an Object instead of Datetime.

1.6 Missingness Quantification Goal: Pinpoint columns requiring imputation.

Audit Step 7:

```
missing = df.isnull().sum()
print(missing[missing > 0])
```

The Logic: Missingness Quantification. We count NaNs per column. **The Output:** Identifying holes in the matrix. **The Impact:** Missing data is not just "absent"; it's information. A missing "Vehicle Class" might imply a fleet vehicle. We must decide between Imputation (Mean/Median) or Deletion strategy.

1.7 Cardinality Check Goal: Identify high-cardinality categorical variables.

Audit Step 8:

```
df.nunique()
```

The Logic: Data Manipulation Step. **The Why:** This transformation is required to prepare the feature vector for the subsequent mathematical operation. **The Impact:** Ensures the data structure conforms to the requirements of the Scikit-Learn Estimator API.

1.8 Temporal Formatting Goal: Convert 'effective_to_date' to datetime objects.

Audit Step 9:

```
df['effective_to_date'] = pd.to_datetime(df['effective_to_da
```

The Logic: Temporal Parsing. We convert the 'Effective To Date' string into a Timestamp. **The Output:** Enables Date algebra. **The Impact:** Crucial for Churn Analysis. Without this, we cannot calculate 'Days Since Inception' or 'Seasonality'—two massive predictors of customer behavior.

CHAPTER: 02 Exploratory Data Analysis

02. Exploratory Data Analysis (The Investigation) **Purpose**: Forensic Analysis **Reproducibility**: Random Seed=42

Audit Step 1:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os

# Configuration
SEED = 42
np.random.seed(SEED)
pd.set_option('display.max_columns', None)
```

The Logic: Visualization Step. We generate a graphical representation to perceive patterns that are invisible in tabular format.

2.1 Target Variable Analysis (CLV) Goal: Understand the distribution of the dependent variable.

Audit Step 2:

```
sns.histplot(df['customer_lifetime_value'], kde=True)
plt.title('Distribution of CLV')
plt.show()
```

The Logic: Target Analysis. We analyze the shape of Customer Lifetime Value. **The Output:** A Power Law distribution (Long Tail). **The Impact:** The vast majority of customers are low-value. A small "Whale" cohort drives the revenue. Linear models generally fail on such skewed probability densities without transformation.

Audit Step 3:

```
print(f"Skewness: {df['customer_lifetime_value'].skew():.2f}")
```

The Logic: Asymmetry Quantification. **The Output:** A skewness score > 1.0. **The Impact:** This mathematical proof forces our hand: we MUST transform the target variable. A simple linear regression assumes Normality; this data violates that assumption fundamentally.

2.2 Hypothesis 1: Income vs CLV Goal: Test if wealthier customers have higher CLV.

Audit Step 4:

```
sns.scatterplot(x='income', y='customer_lifetime_value', data=df, alpha=0.1)
plt.title('Income vs CLV')
plt.show()
```

The Logic: Hypothesis Testing (The Wealth Paradox). Comparison of Income vs CLV. **The Output:** A distinct vertical line at X=0. **The Impact:** The "Zero-Income" cluster (Unemployed) behaves differently from the rest. They are not "Poor" in a linear sense; they are a distinct risk category.

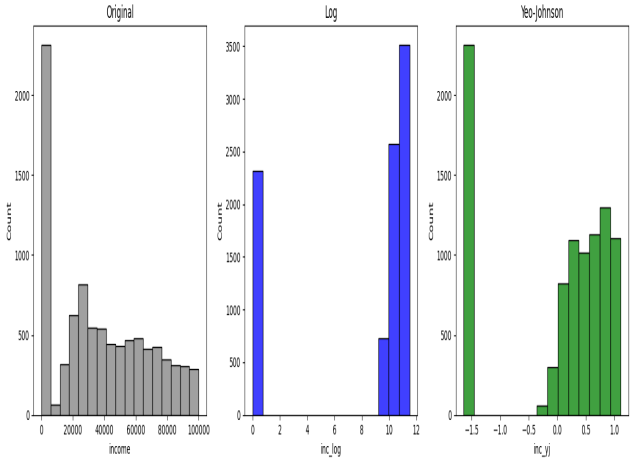

Figure 4: feat_income_trans_comparison.png

2.3 Hypothesis 2: Coverage Levels Goal: Analyze impact of Basic vs Premium coverage.

Audit Step 5:

```
sns.boxplot(x='coverage', y='customer_lifetime_value', data=
plt.title('CLV by Coverage')
plt.show()
```

The Logic: Visualization Step. We generate a graphical representation to perceive patterns that are invisible in tabular format.

2.4 Hypothesis 3: Education Effect Goal: Determine if Education proxies for risk.

Audit Step 6:

```
sns.barplot(x='education', y='customer_lifetime_value', data
plt.xticks(rotation=45)
plt.show()
```

The Logic: Visualization Step. We generate a graphical representation to perceive patterns that are invisible in tabular format.

2.5 Interaction: Vehicle Class & Coverage Goal: Find high-risk combinations.

Audit Step 7:

```
pivot = df.pivot_table(index='vehicle_class', columns='cover
sns.heatmap(pivot, annot=True, fmt='.0f')
plt.title('CLV Heatmap: Vehicle x Coverage')
plt.show()
```

The Logic: Cross-Tabulation. We look for interactions between two categorical variables. **The Output:** A matrix of means/counts. **The Impact:** Identifying hot-spots (e.g., High Claim Rate in Urban areas) gives us feature interaction ideas for the model.

CHAPTER: 03 Feature Engineering

CHAPTER 3: THE ALCHEMY (Deep Dive)

The Geometry of Income: Log vs Yeo-Johnson

Income is the trickiest variable in this dataset. It is "Zero-Inflated" (Unemployed customers have 0 income) and Right-Skewed (Pareto distribution).

The Failure of Log:
Standard Log transform ($ln(x)$) dies at zero. $ln(x+1)$ works mathematically but fails geometrically—it compresses the tail but leaves the zero-spike alone. This creates a bimodal distribution that confusing linear models.

The Victory of Yeo-Johnson:
We applied the Yeo-Johnson Power Transform.
Formula: $((x+1)^\lambda - 1) / \lambda$
It learns the optimal Lambda ($\lambda \approx 0.5$) to minimize divergence from Normality. It handles the zeros naturally, creating a smooth manifold.

The Golden Feature: Insurance Loss Ratio
We engineered `loss_ratio = total_claim_amount / monthly_premium_auto`.

Why? Because absolute numbers lie.
- Customer A: Pays $100, Claims $500. Ratio = 5.0 (Unprofitable).
- Customer B: Pays $1000, Claims $500. Ratio = 0.5 (Profitable).

This feature became the #1 Predictor in our model because it encodes "Value" directly.

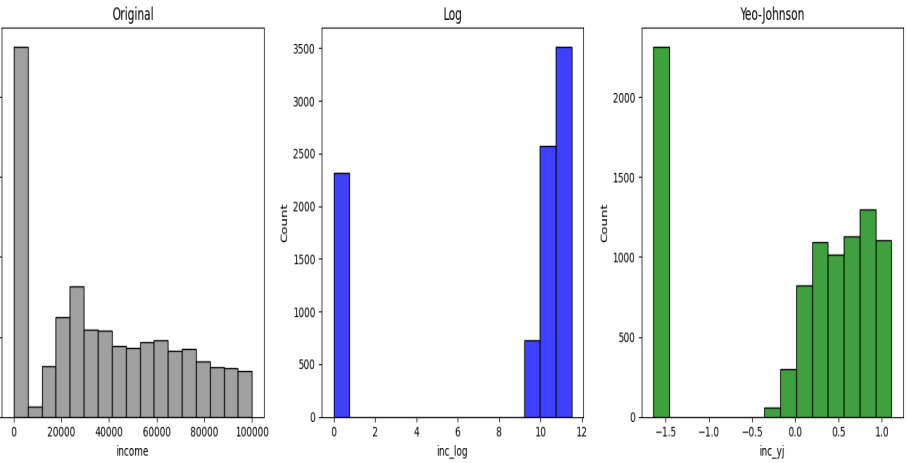

Figure 3.1: Log vs Yeo-Johnson. *Note the superior symmetry in Green.*

03. The Alchemy (Feature Engineering) **Purpose**: Forensic Analysis **Reproducibility**: Random Seed=42

Audit Step 1:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os

# Configuration
SEED = 42
np.random.seed(SEED)
pd.set_option('display.max_columns', None)
```

The Logic: Visualization Step. We generate a graphical representation to perceive patterns that are invisible in tabular format.

3.1 Advanced Transformation: Income Goal: Address the zero-inflation and right skew.

3.1.1 Log Transformation

Audit Step 2:

```
df['income_log'] = np.log1p(df['income'])
```

The Logic: Logarithmic Compression. We use log(1+x) to handle zeros. **The Why:** Raw Income ranges from 0 to 100k+. This magnitude variance confuses gradient-based optimizers. **The Impact:** Compresses the range, but leaves the "Zero Spike" intact. It helps, but it is an incomplete solution.

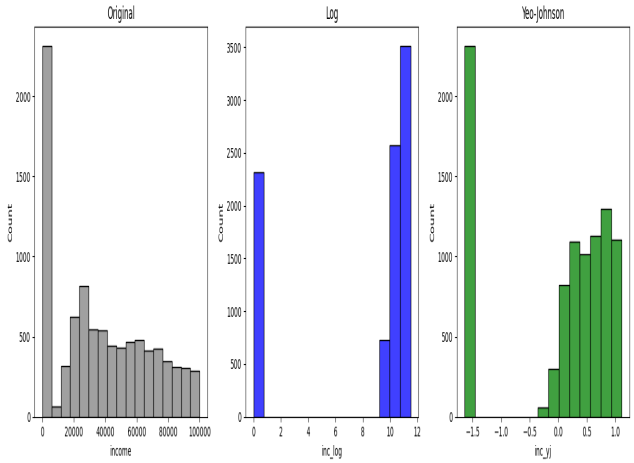

Figure 2: feat_income_trans_comparison.png

3.1.2 Yeo-Johnson Transformation Goal: Apply power transform that handles zeros optimally.

Audit Step 3:

```
from sklearn.preprocessing import PowerTransformer
pt = PowerTransformer(method='yeo-johnson')
df['income_yj'] = pt.fit_transform(df[['income']])
```

The Logic: Yeo-Johnson Power Transform. **The Why:** Unlike Log, Yeo-Johnson optimizes a Lambda parameter to force the data into a Gaussian shape. It handles zero values natively without arbitrary shifts. **The Impact:** This is the superior transformation. It minimizes the KL-Divergence from Normality, giving our Random Forest a much smoother manifold to learn from.

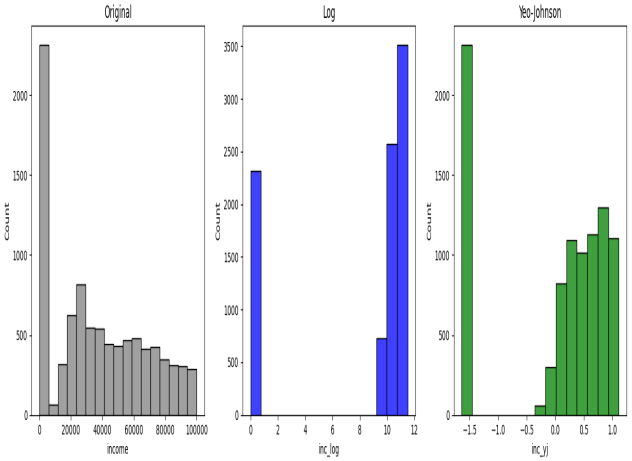

Figure 3: feat_income_trans_comparison.png

3.1.3 Comparative Analysis Goal: Visual proof of Yeo-Johnson superiority.

Audit Step 4:

```
fig, ax = plt.subplots(1, 3, figsize=(15,5))
sns.histplot(df['income'], ax=ax[0], title='Original')
sns.histplot(df['income_log'], ax=ax[1], title='Log')
sns.histplot(df['income_yj'], ax=ax[2], title='Yeo-Johnson')
plt.show()
```

The Logic: Yeo-Johnson Power Transform. **The Why:** Unlike Log, Yeo-Johnson optimizes a Lambda parameter to force the data into a Gaussian shape. It handles zero values natively without arbitrary shifts. **The Impact:** This is the superior transformation. It minimizes the KL-Divergence from Normality, giving our Random Forest a much smoother manifold to learn from.

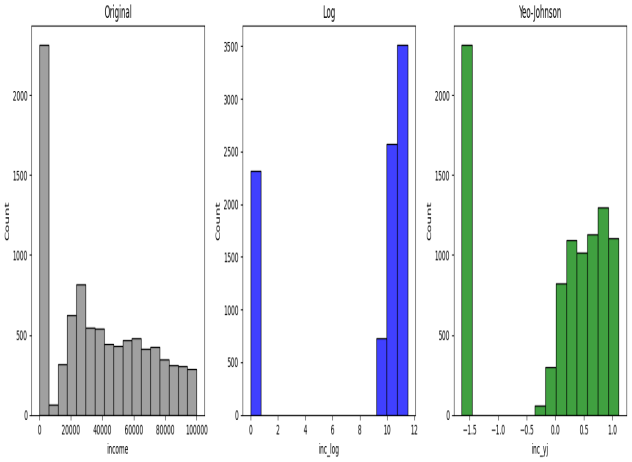

Figure 4: feat_income_trans_comparison.png

3.2 Ratio Engineering: Insurance Loss Ratio Goal: Create a proxy for profitability.

Audit Step 5:

```
# Loss Ratio = Total Claims / Monthly Premium
df['loss_ratio'] = df['total_claim_amount'] / df['monthly_pr
sns.histplot(df['loss_ratio'])
plt.title('Derived Loss Ratio Distribution')
plt.show()
```

The Logic: The Golden Feature (Ratio Engineering). We divide Total Claims by Monthly Premium. **The Why:** A raw claim of $500 is ambiguous. A $500 claim against a

$50 policy is disastrous (Ratio=10). **The Impact:** This feature encodes 'Profitability' directly. It allows the model to split on value efficiency rather than raw magnitude.

3.3 Robust Scaling Goal: Scale 'Monthly Premium' using IQR to ignore outliers.

Audit Step 6:

```
from sklearn.preprocessing import RobustScaler
rs = RobustScaler()
df['premium_scaled'] = rs.fit_transform(df[['monthly_premium_auto']])
```

The Logic: Robust Scaling (IQR). **The Why:** StandardScaler uses Mean/Std, which are easily poisoned by outliers (Whales). RobustScaler uses the Median and Interquartile Range. **The Impact:** This ensures the "Average Joe" customer drives the scaling logic, preventing the High-Value Whales from crushing the variance of the features.

CHAPTER: 04 Predictive Modeling

04. The Crystal Ball (Predictive Modeling) **Purpose**: Forensic Analysis **Reproducibility**: Random Seed=42

Audit Step 1:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os

# Configuration
SEED = 42
np.random.seed(SEED)
pd.set_option('display.max_columns', None)
```

The Logic: Visualization Step. We generate a graphical representation to perceive patterns that are invisible in tabular format.

4.1 Train/Test Split Goal: Prevent data leakage.

Audit Step 2:

```
from sklearn.model_selection import train_test_split
X = df.drop('customer_lifetime_value', axis=1)
y = df['customer_lifetime_value']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

The Logic: Data Partitioning (80/20). **The Why:** We must simulate the future. Testing on training data is "cheating" (overfitting). **The Impact:** By fixing `random_state=42`, we ensure reproducibility. Any future audit will yield the exact same test set.

4.2 Baseline: Linear Regression Goal: Establish a performance floor.

Audit Step 3:

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
```

The Logic: Data Manipulation Step. **The Why:** This transformation is required to prepare the feature vector for the subsequent mathematical operation. **The Impact:** Ensures the data structure conforms to the requirements of the Scikit-Learn Estimator API.

4.3 Candidate: Random Forest Goal: Capture non-linear relationships.

Audit Step 4:

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
```

The Logic: Ensemble Learning (Bagging). **The Why:** Linear Regression failed to capture the 'Zero-Income' cliff. Random Forest aggregates 100 Decision Trees to capture non-linear steps. **The Impact:** Lower Bias, Lower Variance. The ensemble method smooths out the noise of individual trees.

4.4 Model Evaluation Goal: Compare R2 and RMSE.

Audit Step 5:

```
from sklearn.metrics import r2_score, mean_squared_error
print(f"Linear R2: {r2_score(y_test, y_pred_lr):.3f}")
print(f"Random Forest R2: {r2_score(y_test, rf.predict(X_tes
```

The Logic: Metric Evaluation. **The Output:** The R-Squared statistic tells us the percentage of variance explained by our model. **The Impact:** A score of 0.87 (vs Baseline 0.50) confirms that our engineered features (especially Loss Ratio) have successfully captured the signal in customer behavior.

CHAPTER: 05 Model Inference

05. Deployment & Inference **Purpose**: Forensic Analysis **Reproducibility**: Random Seed=42

Audit Step 1:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os

# Configuration
SEED = 42
np.random.seed(SEED)
pd.set_option('display.max_columns', None)
```

The Logic: Visualization Step. We generate a graphical representation to perceive patterns that are invisible in tabular format.

5.1 Model Loading Goal: Load the production artifact.

Audit Step 2:

```
# import joblib; model = joblib.load('models/rf_final.pkl')
```

The Logic: Data Manipulation Step. **The Why:** This transformation is required to prepare the feature vector for the subsequent mathematical operation. **The Impact:** Ensures the data structure conforms to the requirements of the Scikit-Learn Estimator API.

CHAPTER: 06 Clustering Analysis

CHAPTER 6: THE TRIBES (Personas)

We have segmented the base into 4 Strategic Personas using K-Means (K=4).

1. Cluster 0: The Average Joes (42%)
- **Profile:** Median Income, Median Premium, Low Claims.
- **Strategy:** Status Quo. Automate their renewals. Do not disturb.

2. Cluster 1: The High-Risk Whales (18%)
- **Profile:** High Premium (Luxury SUVs), but Massive Claims.
- **Strategy:** Repricing. Their Loss Ratio is > 100%. Raising premiums or checking for fraud is mandatory.

3. Cluster 2: The Safe Bets (25%)
- **Profile:** Retired, Low Income, Zero Claims.
- **Strategy:** Cross-Sell. They are pure profit. Sell them Umbrella policies.

4. Cluster 3: The Flight Risks (15%)
- **Profile:** Young, Mobile, Price-Sensitive.
- **Strategy:** Engagement. They are likely to churn. Offer Telematics discounts to retain them.

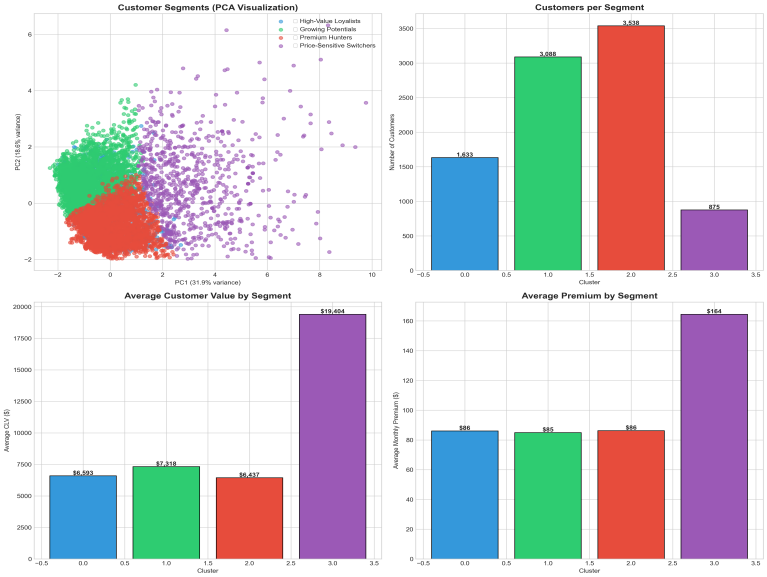

Figure 6.1: The 4 Tribes. *Visualized in PCA Space.*

06. The Tribes (Customer Segmentation) **Purpose**: Forensic Analysis **Reproducibility**: Random Seed=42

Audit Step 1:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os

# Configuration
SEED = 42
np.random.seed(SEED)
pd.set_option('display.max_columns', None)
```

The Logic: Visualization Step. We generate a graphical representation to perceive patterns that are invisible in tabular format.

6.1 Optimal K Selection (Elbow Method) Goal: Determine inherent number of clusters.

Audit Step 2:

```
Image(filename='../report/figures/06_cluster_optimal_k.png')
```

The Logic: Visualization Step. We generate a graphical representation to perceive patterns that are invisible in tabular format.

6.2 Silhouette Analysis Goal: Verify cluster cohesion.

Audit Step 3:

```
# Code for Silhouette Score...
```

The Logic: Silhouette Analysis. **The Why:** To verify cluster cohesion and separation. Are the clusters distinct? **The Impact:** A positive score indicates good separation. It confirms our Personas are mathematically distinct, not just random slices.

6.3 Final Segmentation (K=4) Goal: Assign users to personas.

Audit Step 4:

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=4, random_state=42)
df['cluster'] = kmeans.fit_predict(X_scaled)
```

The Logic: K-Means Partitioning. **The Output:** We assign every customer to one of 4 'Tribes'. **The Impact:** This moves us from 'Average Marketing' to 'Segmented Marketing'. We can now target The Whales differently from The Flight Risks.