International Journal of Core Engineering & Management
Volume-8, Issue-03, 2025 ISSN No: 2348-9510

MACHINE LEARNING PIPELINE FOR AUTOMOTIVE PROPENSITY MODELS

Vaibhav Tummalapalli
Atlanta, USA
Vaibhav.tummalapalli21@gmail.com

Abstract

The automotive industry increasingly relies on machine learning (ML) to drive personalized marketing, customer retention, and predictive analytics. This paper introduces a comprehensive ML pipeline designed for automotive propensity models. It outlines a systematic process from data partitioning and feature selection to model evaluation and deployment. The pipeline assumes that the data is preprocessed with feature aggregation and aims to deliver accurate, interpretable, and scalable ML solutions for automotive applications

Index Terms – ***Machine Learning, Imputation, Outlier Detection, ML interpretability Hyperparameter Tuning, Data Cleaning, Preprocessing, Feature Engineering, Transformations, Deployment & Monitoring***

I. INTRODUCTION

Propensity models in the automotive domain play a pivotal role in predicting customer behaviors such as service visits, vehicle purchases, and campaign engagement. Developing such models involves a robust ML pipeline that ensures data readiness, model stability, and business relevance. This paper presents an end-to-end ML pipeline tailored for automotive propensity modeling, addressing critical steps from data preparation to post-deployment monitoring.

II. DATA PROCESSING FOR MACHINE LEARNING

A. Data Partition

To ensure robust model development and evaluation, the dataset is divided into three distinct subsets: training, validation, and testing. The specific partition percentages depend on the size of the dataset and the available processing power [2]. When dealing with millions of data points and sufficient computational resources, a higher percentage of the data is allocated to the training set (e.g., 90% training, 10% validation). This allows the model to learn effectively from a substantial volume of data while still leaving enough for validation. For more limited datasets, common splits include 70-30 (training-validation) or 60-40. These splits balance the need for a sufficient training sample while ensuring adequate data is available for validation and testing

Purpose of Each Subset:

- **Training Set**: Used to train the model and optimize its parameters.
- **Validation Set**: Helps tune hyperparameters and assess the model's performance during development.
- **Testing Set**: Reserved for evaluating the model's generalizability on unseen data.

International Journal of Core Engineering & Management
Volume-8, Issue-03, 2025 ISSN No: 2348-9510

This structured approach ensures the model is trained, validated, and tested rigorously, minimizing the risk of overfitting and ensuring stable performance

B. Basic Data Cleaning

The goal is to remove irrelevant, redundant, or noisy data that may hinder model performance.

Column Selection:

- **Low Variance Columns:** Drop columns with a single value or near-zero variance, as they do not contribute to distinguishing target outcomes.

- **High Missing Percentage:** Discard columns with missing values exceeding a threshold (e.g., 30%-50%), depending on data quality. For well-populated datasets, use a lower or stricter threshold (e.g. 30%), whereas for sparsely populated datasets, adopt a higher threshold (e.g. 50%) to retain sufficient columns for analysis.

Row Cleaning: Remove duplicate entries to prevent skewing model results

C. Categorical Data Handling

Categorical data requires careful preprocessing to ensure that the information is represented in a format suitable for machine learning algorithms, which generally work with numeric inputs. Below are the key strategies for handling categorical data effectively:

Encoding Methods Based on Data Nature

- **Ordinal Encoding**: Used for ranked categories where there is an inherent order. Example: For a "Customer Satisfaction" column with levels like *Low, Medium, High*, ordinal encoding maps them to numerical values (e.g., Low = 1, Medium = 2, High = 3). Preserves the order of the categories, which is critical for maintaining the meaning of the data.

- **One-Hot Encoding**: Suitable for non-ordinal categories where there is no natural ranking. Creates a binary column for each category. For example, for a "Car Type" column with categories like *SUV, Sedan, Truck*, one-hot encoding creates three new columns (*Car Type SUV*, *Car Type Sedan*, *Car Type Truck*) with binary indicators. Ideal for categories with fewer levels, as it avoids an explosion in the number of features.

- **Target or Frequency Encoding**: Maps categories to their mean target variable value or frequency in the data. Example: For a "Region" column, assign each region the average response rate of customers from that region. Particularly useful for high-cardinality categorical variables.

Handling High-Cardinality Categorical Variables

- **Combining Levels**: Categories with low frequencies or small event rates can be grouped together. Use domain knowledge or event rate thresholds to decide on groupings. Example: Combine rare car colors like *Pink, Maroon, and Turquoise* into a single category called *Other Colors*. Reduces sparsity in the data and prevents overfitting in the model.

Preprocessing Considerations:

- **Dummy Variable Trap**: When using one-hot encoding, drop one column to avoid multicollinearity. For example, if encoding *SUV, Sedan, Truck*, retain only two columns (e.g., *SUV* and *Sedan*) and infer the third category when both are 0.
- **Maintaining Interpretability**: Choose encoding methods that align with the end goal. For example, one-hot encoding may be better for models requiring high interpretability, while target encoding may be more effective for boosting model performance

III. OUTLIER DETECTION

Outliers are extreme values in the data that can distort the analysis and adversely affect the performance of machine learning models. Effective detection and treatment of outliers are essential for ensuring data quality and stability in predictive models. The choice of methods for outlier detection depends on the nature of the data distribution and domain knowledge

A. Normal & Gaussian Distribution

- For features that follow a normal distribution, statistical measures like mean and standard deviation are effective in detecting outliers.

- **Cap and Floor with Standard Deviation Method**:
  - Calculate the mean (μ) and standard deviation (σ) of the feature.
  - Define outliers as values lying outside the range: $[\mu - 3\sigma, \mu + 3\sigma]$

  Example: For a column representing *customer age*, if the mean is 40 years with a standard deviation of 10 years, any value below 10 or above 70 is capped or floored to the nearest boundary

.

B. Non-Gaussian Distribution

For features that do not follow a normal distribution, robust methods like IQR (Interquartile Range) or MAD (Median Absolute Deviation) are used.

Interquartile Range (IQR) Method:

- Compute the first quartile (Q1) and third quartile (Q3).
- Define the IQR as: $IQR = Q3 - Q1$
- Identify outliers as values lying outside the range: $[Q1 - 1.5 \times IQR, Q3 + 1.5 \times IQR]$
- Example: For a feature representing *transaction amount*, if Q1 = \$100 and Q3 = \$500, any value below \$-50 or above \$650 is treated as an outlier.

Median Absolute Deviation (MAD):

- Compute the median (M) and the absolute deviation of each value from the median.
- Define the MAD as: $MAD = \text{median}(|x_i - M|))$
- Use a multiplier (e.g., 3 or 3.5) to define thresholds for outliers: $[M - k \times MAD, M + k \times MAD]$

Domain-Specific Limits

For certain features, outlier limits can be determined using domain expertise. Example: For a payment field, if the maximum allowable transaction value is $10,000, any value exceeding this can be capped at $10,000, and negative values (if invalid) can be set to zero. This approach is particularly useful for variables with predefined constraints, such as geographic coordinates, speed limits, or inventory levels

Practical Implementation

- **Capping and Flooring:** Replace extreme values with the nearest valid value (boundary). This helps preserve the dataset size and ensures that the outliers do not skew the analysis.

- **Influential Rows:** Use row-level detection methods like Cook's Distance for linear or logistic regression models. Cook's Distance measures the influence of each observation on the model coefficients. If the cook's statistic is greater than the Threshold ($D > 4/n$ where n is the number of observations in the data). Observations with a high Cook's Distance should be reviewed and potentially removed.

- **Visualization:** Use boxplots or scatter plots to visually inspect outliers. This is especially useful for identifying patterns, clusters, or anomalies that may indicate systematic errors or unusual behavior.

- **Iterative Refinement:** After outlier treatment, reassess the data distribution. For Gaussian variables, recheck if the distribution remains normal post-treatment. If the data remains heavily skewed, consider applying transformations like logarithmic or square root adjustments

IV. IMPUTATION

Missing data is a common challenge in data preprocessing, and selecting the appropriate imputation method is crucial for preserving data quality and ensuring model reliability. The choice of technique depends on the extent of missing data, the nature of the variable (categorical or numeric), and the downstream modeling requirements.

Low Missing Rates (5%-10%)

When the percentage of missing values is relatively small, simpler imputation techniques can be applied effectively:

- **Mean Imputation**: Replace missing values with the column mean. This is best suited for continuous variables with symmetric distributions, as it preserves the overall mean of the dataset.
- **Median Imputation**: Replace missing values with the column median. This approach is robust to outliers and is preferred for skewed distributions.
- **Mode Imputation**: For categorical variables, replace missing values with the most frequently occurring category. This method is simple and effective when the mode

represents a meaningful category.

High Missing Rates
For variables with a high proportion of missing values, more advanced techniques are needed to avoid introducing bias or oversimplification:

K-Nearest Neighbors (KNN) Imputation

- Imputes missing values based on the similarity of other rows in the dataset. The value is calculated as a weighted average of the k-nearest neighbors. This captures relationships between features and works well with both numeric and categorical data (after encoding). The challenge with this technique is that it is computationally expensive, especially for large datasets and sensitive to the choice of k (number of neighbors) and distance metric. Suitable when sufficient data points are available for reliable neighbor identification.

Multivariate Imputation by Chained Equations (MICE)

- **Concept**: Uses regression models to predict missing values based on other variables in the dataset. The process is iterative, imputing missing values for one variable at a time, conditioned on the other variables. Accounts for relationships between variables, resulting in more accurate imputations. Effective for complex datasets with intricate relationships between features.

Weight of Evidence (WOE) Imputation

- **Concept**: Bins numeric variables, creates a "missing" bin for null values, and assigns each bin a WOE value based on the target variable.

- **Steps to computing WOE**:
  1. Divide the variable into bins (e.g., quartiles or domain-specific ranges).
  2. Calculate WOE for each bin

$$WOE = \ln\left(\frac{\text{Proportion of events in the bin}}{\text{Proportion of non-events in the bin}}\right)$$

  3. Replace missing values with the WOE value of the "missing" bin.
- **Advantages of WOE**:
  - Retains the predictive power of the variable by linking imputation to the target variable. Easily interpretable in logistic regression and other models that use WOE as input. This is not suitable if the variable has no clear relationship with the target.

Best Practices: Assess the impact of imputed values on model performance to ensure the method aligns with the problem context. Create binary flags for missing values before imputation. These flags can serve as additional features, capturing information about missingness patterns. Experiment with different imputation methods and assess their impact on both training and validation performance. By tailoring imputation techniques to the characteristics of the data, businesses can maintain data integrity and enhance the predictive power of their models**.**

V. FEATURE SELECTION

Feature selection is a critical step in the machine learning pipeline that reduces dimensionality, improves model interpretability, and prevents overfitting by retaining only the most predictive features [1]. This process balances model complexity and performance, ensuring that the selected features contribute significantly to the predictive power of the model.

A. Statistical Techniques:

Statistical methods provide an initial assessment of feature importance and relationships with the target variable [3].

- **Information Value (IV):** Measures the predictive power of a feature in distinguishing between classes (e.g., events vs. non-events). IV values guide feature selection: $IV < 0.02$: Weak predictor. $0.02 \leq IV < 0.1$: Medium predictor. $IV \geq 0.1$: Strong predictor. Features with higher IV values are prioritized for modeling.
- **Chi-Square Test:** Evaluates the independence of categorical variables and the target variable. Higher Chi-Square values indicate stronger relationships with the target variable, helping identify key predictors [3].
- **ANOVA (Analysis of Variance):** Quantifies the variance in the predictor variable explained by target feature. Useful for continuous features, helping isolate variables with significant predictive potential [3].
- **Correlation Coefficients:** Measures the strength and direction of linear relationships between numeric variables. Identifies redundant features by highlighting highly correlated pairs, enabling the removal of less informative variables.

B. Machine Learning-Based Techniques

ML-based methods incorporate the predictive power of features into the selection process.

- **Recursive Feature Elimination (RFE):** Iteratively trains models using subsets of features and removes the least important ones at each iteration. Works well with tree-based algorithms like Decision Trees, Random Forests, and Gradient Boosted Machines, which internally rank features. Produces a ranked list of features, enabling targeted selection.
- **Voting Methods:** Aggregates feature importance rankings from multiple algorithms (e.g., Random Forest, Gradient Boosting, Logistic Regression). Features that consistently appear as important across algorithms are retained, ensuring robustness and reducing model bias.

C. Multicollinearity Checks

Multicollinearity arises when two or more features are highly correlated, leading to redundancy and instability in linear models. Addressing multicollinearity is essential for maintaining model robustness.

- **Variable Clustering:** Groups features based on their pairwise correlations. For each cluster, the most representative feature (e.g., with the highest IV or lowest p-value) is selected, eliminating redundancy.
- **Variance Inflation Factor (VIF):** Quantifies the extent of multicollinearity. $VIF = 1$: No multicollinearity. $VIF > 5$: Moderate multicollinearity (consider removal). VIF >

10: Severe multicollinearity (remove feature). Features with high VIF values are sequentially removed, ensuring a well-conditioned feature set.

Best Practices

- Feature selection is rarely a one-time step. Iteratively evaluate selected features using cross-validation to ensure the best subset for the model.
- Incorporate business expertise to prioritize features with known relevance to the problem (e.g., vehicle type in automotive modeling). In addition to improving predictive performance, strive to retain features that provide actionable insights for stakeholders.
- Removing too many features may degrade the model's ability to generalize, especially when working with complex relationships

VI. DATA TRANSFORMATIONS

Data transformation is a critical step in preparing raw data for machine learning models, ensuring that features are properly scaled, normalized, and encoded to enhance model performance and interpretability. This step addresses data irregularities such as skewness, scaling issues, and high-cardinality categorical variables, enabling the model to effectively learn patterns.

A. Numerical Features

Numeric transformations are applied to address issues such as skewness, non-linearity, and over-dispersion in continuous variables.

- **Normalization of Skewed Data:** Skewed data can bias model training and predictions. Apply transformations to reduce skewness. Log Transformation is useful for right-skewed variables, e.g., service cost or vehicle mileage. Square Root Transformation moderates the effect of extreme values while maintaining data structure whereas power Transforms adjusts skewness by applying exponents or fractional powers.
- **Optimal Binning:** Discretize continuous variables into bins based on the relationship with the target variable. Use methods like Chi-Square binning to create optimal cutoffs that maximize the predictive power of each bin. Example: "Service spending" can be binned into low, medium, and high categories, improving interpretability and robustness.

B. Standardization and Scaling

Standardization ensures that all features contribute equally to the model, particularly in algorithms sensitive to feature scaling.

- **Standardization:** Adjust variables to have a mean of 0 and a standard deviation of 1. Example: For features like vehicle price and customer spending, standardization prevents larger numerical values from dominating the model's optimization process.

- **Scaling:** Scale values to a fixed range, typically [0,1], for algorithms like neural networks that perform better with bounded inputs. Example: Scale "vehicle mileage" and "number of services" to ensure consistency in feature magnitude.

C. Best Practices

- **Select Transformations Based on Model Requirements.** For linear models, focus on normalizing and scaling; for tree-based models, transformations may not be necessary.

- **Preserve interpretability** by balancing transformation complexity with the ability to explain feature relationships to stakeholders. Experiment with different transformations and evaluate their impact on model performance.

- **Leverage Domain Knowledge** and use business insights to guide binning thresholds, encoding strategies, and scaling ranges

VII. MODEL DEVELOPMENT & EVALUATION

The modeling phase is the cornerstone of the machine learning pipeline, where the focus is on selecting, training, and optimizing algorithms to predict target outcomes effectively. This phase balances robustness, accuracy, and interpretability to create a predictive model that meets both technical and business objectives.

Model Selection

The model selection process begins with simpler models to establish a baseline and gradually progresses to more complex algorithms for enhanced predictive performance.

- **Baseline Models:** Start with interpretable algorithms such as Logistic Regression or Decision Trees. For example, in an automotive churn prediction model, logistic regression can help identify the most influential factors leading to customer attrition.
- **Advanced Algorithms:** For improved accuracy and handling of non-linear relationships, use Neural networks [4] or any of the advanced tree-based algorithms like Random Forests, XG Boost, Light GBM, or Cat Boost.

Hyperparameter Tuning

Hyperparameter tuning is essential for optimizing model performance by fine-tuning algorithm settings to balance bias and variance [7].

- **Grid Search:** Systematically evaluates combinations of hyperparameters using cross-validation to identify the best configuration. Example: Tuning the maximum depth and learning rate in a GBM to find the optimal trade-off between model complexity and accuracy.
- **Random Search:** Samples random combinations of hyperparameters for quicker evaluation when the search space is large.
- **Bayesian Optimization:** Uses probabilistic models to explore hyperparameter space efficiently, focusing on high-performing regions. Example: Optimizing hyperparameters of a neural network, such as learning rate and the number of layers, using Bayesian techniques for faster convergence.

- **Stacking:** Combines multiple models (e.g., linear models and GBMs) by using their predictions as inputs to a meta model. Example: Stacking logistic regression and GBMs to predict vehicle service propensities, leveraging the interpretability of one model and the accuracy of another.

Best Practices

- **Start Simple:** Begin with interpretable models to understand feature relationships and establish a baseline before transitioning to complex algorithms.
- **Iterative Approach:** Test and compare multiple algorithms using consistent evaluation metrics to select the best-performing model.
- **Balance Accuracy and Interpretability:** Ensure that the selected model aligns with the business's ability to interpret and act on its predictions.

Monitor Overfitting: Use techniques like cross-validation and regularization to prevent overfitting, especially in ensemble methods

Model Metrics

Ranking Metrics: Ranking metrics are essential for marketing and propensity models where the objective is to prioritize customers based on their likelihood to respond or act. Key metrics include [8]:

- **Lift:** Measures the improvement in the response rate of the top-ranked customers compared to a random selection. Example: In a vehicle service retention model, a lift of 3 for the top decile indicates that customers in the top 10% are three times more likely to return for service than the general population.
- **Capture Rate:** Indicates the percentage of total positive responses captured within a specific segment (e.g., top deciles). Example: If the top three deciles capture 65% of responders in a purchase propensity model, it demonstrates effective targeting.
- **KS Statistic (Kolmogorov-Smirnov):** Quantifies the maximum separation between the cumulative distributions of responders and non-responders across score thresholds. Interpretation: A higher KS value (typically >40) signifies better model discrimination.

Performance Validation: Performance validation ensures that the model remains stable across datasets and can generalize well with new data.

- **Training vs. Validation Metrics:** Compare performance metrics (e.g., Lift, KS, accuracy) across training and validation datasets to detect overfitting. Example: If the lift in training is 4.0 but drops to 2.5 in validation, the model may be overfitting to the training data.
- **Back-Test Validation:** Set aside a portion of data (e.g., a separate period) as a back-test dataset to simulate real-world performance. Example: In an automotive sales model, back-testing using data from a recent sales period can verify that the model captures current trends effectively.

By focusing on ranking metrics, performance validation, and real-world testing, this evaluation framework ensures the development of robust, generalizable models. In the automotive industry, these insights directly enhance marketing strategies, customer retention efforts, and operational decision-making.

Interpretability

Provost and Fawcett [5] emphasize the importance of business relevance, which is central to the end-to-end design of this pipeline. Interpretability, as stressed by Verbeke et al. [6], is a key requirement for marketing models used by decision-makers in dealerships.

- **SHAP Values:**
  - **Purpose:** Quantify the contribution of each feature to individual predictions, enhancing transparency. **Example:** In a service propensity model, SHAP values can show that "months since last service" is a stronger predictor than "service type diversity."
  - **Visualization:** SHAP summary plots rank features by importance and provide insights into their positive or negative impacts on predictions. Helps stakeholders understand why a customer is flagged as high propensity, enabling trust and better decision-making.
- **Partial Dependence (PD) Plots:**
  - **Purpose:** Visualize the relationship between specific features and the target variable while holding other features constant. **Example:** A PD plot for "spend in the last 12 months" in a purchase propensity model may show that higher spending reduces the likelihood of immediate purchase.
  - **Use Case:** Identifies thresholds or ranges for feature values that are most influential, enabling better segmentation and targeted interventions.

Back Testing

The objective is to validate the model's performance on unseen data, ensuring its generalizability and stability. **Data Partitioning for Back Testing:** Reserve a portion of the dataset (e.g., a specific time or customer cohort) that was not used during training or validation. Apply the model on the back test data and calculate ranking metrics (Lift, Capture Rate, KS Statistic) and classification metrics (Precision, Recall, F1 Score) on back-test data. Compare these metrics with those from training and validation datasets. **These** metrics should be comparable across datasets, indicating stability. **Example:** A purchase propensity model achieving 3.5x Lift in training and 3.4x in back-testing confirms generalizability. This Ensures confidence in deploying the model to production. Highlights any potential overfitting or underperformance, allowing for pre-deployment adjustments.

Deployment & Monitoring

The goal is to implement the model in production while ensuring its continued relevance and effectiveness over time.

- **Deployment:** Save the trained model. Ensure preprocessing, transformations, and feature engineering steps applied during production match those used in model development. Apply the saved model to score new data, generating actionable predictions.
- **Model Monitoring/Data Drift and Model refresh:** Periodically evaluate model metrics (e.g., Lift, KS, Accuracy) to ensure the model remains effective. Monitor for changes in data distributions or consumer behavior patterns that deviate from training data. Example: If "months since the last service" has an increasing average, this may indicate shifting customer engagement trends. Retrain the model when significant drift or performance degradation is detected.

- **Business Impact:** Ensures long-term model reliability and relevance. Reduces the risk of poor decisions due to outdated models or unanticipated changes in customer behavior

VIII. CONCLUSION

This paper presents a comprehensive ML pipeline tailored for propensity modeling in the automotive industry. Each stage, from data preparation and feature engineering to deployment and monitoring, is designed to ensure robust, interpretable, and actionable models. By incorporating advanced techniques like SHAP values for interpretability and thorough monitoring practices, the framework effectively addresses real-world challenges such as feature importance, model drift, and ensuring consistent performance across diverse datasets. This pipeline equips businesses with the tools needed to make data-driven decisions and optimize customer engagement strategies.

REFERENCES

1. T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd ed. New York: Springer, 2009
2. Tummalapalli Vaibhav. (2025). Stratified sampling in Cohort-based data for Machine learning Model development. International Scientific Journal of Engineering and Management. 04. 1-8. 10.55041/ISJEM03377
3. V.Tummalapalli and K. Konakalla, "Statistical Techniques for Feature Selection in Machine Learning Models," International Journal for Innovative Research in Multidisciplinary Pursuit and Studies (IJIRMPS), vol. 13, no. 3, pp. 1-8, 2025, doi: 10.37082/IJIRMPS.v13.i3.232566
4. G. James, D. Witten, T. Hastie, and R. Tibshirani, An Introduction to Statistical Learning with Applications in R. New York: Springer, 2013.
5. A. Luckow et al., "Deep Learning in the Automotive Industry: Applications and Tools," arXiv preprint arXiv:1705.00346, 2017.
6. F. Provost and T. Fawcett, Data Science for Business: What You Need to Know About Data Mining and Data-Analytic Thinking, O'Reilly Media, 2013
7. Tummalapalli, V. (2025). Outlier detection & treatment for machine learning models. International Journal of Innovative research & Creative Technology, 11(3), 1–8. https://doi.org/10.5281/zenodo.16500050.