Updated Report

Comprehensive Report: Improved Car Insurance Claim Prediction Pipeline

Executive Summary

This report details the enhanced machine learning pipeline for predicting car insurance claims using the Car_Insurance_Claim.csv dataset. The target is the binary OUTCOME (0 = No Claim, 1 = Claim). The pipeline includes data loading, EDA, preprocessing, feature engineering, model training (Logistic Regression, Random Forest, CatBoost), evaluation, feature importance, leaderboard, and a Gradio app for deployment.

Compared to the first version, this iteration adds logging, modular functions, feature importance visualization, a leaderboard, and Gradio app. It reduces code repetition, improves reproducibility, and enhances user interaction. All artifacts (plots, models, metrics, results) are saved for analysis.

The script uses libraries like Pandas, Scikit-learn, CatBoost, and Gradio. It emphasizes reproducibility with a fixed random state (42) and stratified splitting.

1. Environment Setup and Library Imports

Steps Taken:

- **Logging Setup**: Stateful logging to pipeline.log with file and console handlers.
- **Core Imports**:
    - Pandas (pd) and NumPy (np) for data manipulation.
    - Seaborn (sns) and Matplotlib (plt) for visualizations.
    - Scikit-learn modules for preprocessing and tuning.
    - Models: LogisticRegression, RandomForestClassifier, CatBoostClassifier.
    - Evaluation: Metrics and plotting utilities.
    - Imbalanced handling: SMOTE.
- **Constants**: RAW_PATH, RANDOM_STATE = 42, TARGET_COL = "OUTCOME".

Purpose:

Establishes a reproducible environment with logging for traceability.

2. Data Loading and Initial Exploratory Data Analysis (EDA)

Steps Taken:

- **Load Raw Data**: df_raw = pd.read_csv(RAW_PATH), with a working copy df = df_raw.copy().
- **Quick Report Function**: Generates shape, head, dtypes, missing values, target distribution, numeric/categorical summaries, unique values, and top categories.
- **Visualizations**: Boxplots and histograms for numeric columns vs. target, saved to plots/eda.

Key Findings:

- Imbalance in target: 68.67% no claims, 31.33% claims.
- Missing values limited; handled later.
- Categorical skew (e.g., RACE mostly 'majority').
- Potential outliers in numerics.

Purpose:

EDA identifies issues (missing values, imbalance, outliers) and informs preprocessing.

3. Data Splitting

Steps Taken:

- Stratified split: 80/20 ratio, stratify on target, random_state=42.
- Shapes: Train (8000 rows), Test (2000 rows).
- Backup raw train data: X_train_raw = X_train.copy().

Purpose:

Early split prevents leakage; stratification preserves class distribution.

4. Data Preprocessing

Steps Taken:

- **Clean Column Names**: Lowercase, strip spaces, replace with underscores.
- **Standardize Categoricals**: Trim whitespace, lowercase strings.
- **Drop Unnecessary Columns**: 'id', 'race', 'postal_code'.

- **Outlier Clipping**: IQR-based clipping for numeric columns.
- **Impute Missing Values**: Median for numerics, most frequent for categoricals.
- **Binary Consistency**: Map married and children to "Yes"/"No"; keep vehicle_ownership numeric.

Key Changes:

- No more missing values post-imputation.
- Outliers mitigated to prevent bias.
- Dropped columns reduce noise.

Purpose:

Clean data ensures consistency and reduces errors in modeling.

5. Feature Engineering

Steps Taken:

- Create risk_score: Sum of speeding_violations, duis, past_accidents (if present).
- Drop the component columns.

Key Findings:

- risk_score consolidates driving history, improving interpretability.

Purpose:

Engineered features capture domain knowledge, boosting predictive power.

6. Model-Specific Data Preparation

Steps Taken:

- **CatBoost**: Use unencoded X_train_cat, X_test_cat with cat_features.
- **RF/LogReg**: OneHotEncode categoricals, concatenate with numerics to create X_train_encoded, X_test_encoded.

Purpose:

Prepares data for models requiring encoding (RF/LogReg) while allowing CatBoost to handle categoricals natively.

7. Model Evaluation Framework

Steps Taken:

- Define evaluate_model function:
    - Fits model, predicts, computes metrics (accuracy, precision, recall, F1, specificity, ROC-AUC).
    - Generates plots: Confusion matrix, ROC curve.
    - Appends results to a list.

Purpose:

Standardizes evaluation across experiments, focusing on imbalanced-class metrics.

8. Model Experiments

Experiments use evaluate_model with variants for imbalance handling and tuning. Plots/results saved per model type.

8.1 Random Forest

- **Data**: Encoded (X_train_encoded, etc.).
- **Variants**:
    1. Baseline: Default RF.
    2. SMOTE (Untuned): Oversample minority class.
    3. Tuned (No SMOTE): RandomizedSearchCV on params.
    4. SMOTE + Tuned: Same search on SMOTE data.
- **Summary**: DataFrame of metrics; bar chart comparison.

8.2 Logistic Regression

- **Data**: Encoded + Scaled numerics (StandardScaler).
- **Variants**:
    1. Baseline: Default (saga solver, max_iter=1000).
    2. Weighted: class_weight='balanced'.
    3. Tuned (No Weights): RandomizedSearchCV on C, penalty, solver.
    4. Weighted + Tuned: Same with weights.
- **Summary**: Similar to RF.

8.3 CatBoost

- **Data**: Unencoded cleaned data, with cat_features.
- **Variants**:
    1. Baseline: Default.
    2. Weighted: class_weights=[1,2].
    3. Tuned (No Weights): RandomizedSearchCV on depth, learning_rate, etc.
    4. Weighted + Tuned: Same with weights.
- **Summary**: Similar to others.

Key Findings:

- Tuning and imbalance handling improve F1/Recall.
- CatBoost outperforms others (e.g., higher ROC-AUC ~0.88).

Purpose:

Compare algorithms and variants to select the best (CatBoost Weighted + Tuned).

9. Leaderboard and Final Comparison

Steps Taken:

- Select best from each: Tuned variants.
- Build DataFrame: Metrics + rank by Test F1.
- Dynamic bar plot: All metrics across models (saved as leaderboard_comparison_dynamic.png).

Key Findings:

- CatBoost tops leaderboard, balancing precision/recall.

Purpose:

Summarizes experiments for decision-making.

10. Model Persistence

Steps Taken:

- Save models: Joblib to saved_models (e.g., catboost_weighted_tuned.pkl).

- Save results: CSVs in results (per model type).

Purpose:

Enables reuse/deployment without retraining.

11. Deployment with Gradio

Steps Taken:

- Load best CatBoost and metrics.
- Define prediction function: Takes inputs, builds DataFrame, predicts outcome/probability.
- Interface: Dropdowns/sliders for features; displays prediction and metrics.
- Launch: Shareable web app.

Purpose:

Provides an interactive tool for real-world use.

12. Conclusion and Recommendations

The pipeline is robust, covering end-to-end ML workflow. Strengths: Thorough EDA, imbalance handling, comprehensive experiments. Improvements: Add XGBoost experiments; cross-validation in baseline; SHAP for interpretability; handle more edge cases in Gradio.

All steps prioritize reproducibility and visualization. Final model (CatBoost) is production-ready with ~0.73 F1 on test. For deployment, monitor drift and retrain periodically.