MACHINE LEARNING PROJECT 1

A project report on

Enhanced Safe Driver Prediction Challenge

Task 2: Kaggle Submission

CSE 472 Introduction to Machine Learning

Sagar Lekhraj

ERP: 29325

Department of Computer Science

Submitted to: **Dr. Sajjad Haider, PhD**

IBA Karachi

Department of Computer Science

Abstract — Enhanced Safe Driver Prediction Challenge

The Enhanced Safe Driver Prediction Challenge focuses on developing a robust machine learning model to predict the probability that an auto insurance policyholder will file a claim. Using an improved version of the classic Porto Seguro Safe Driver dataset, participants are tasked with maximizing predictive performance measured by the Area Under the Receiver Operating Characteristic Curve (AUROC).

The challenge emphasizes smart feature engineering, effective handling of imbalanced data, and robust model development to achieve superior discriminative capability. Participants must preprocess and analyze high-dimensional insurance data, identify key risk indicators, and optimize model generalization across unseen samples. The ultimate goal is to design a reliable, high-performing classifier that can accurately assess insurance claim risk, contributing to more efficient and data-driven decision-making in the auto insurance industry.

Keywords: Machine Learning, Pattern Recognition, Classification, Supervised learning, Data Engineering, Data Featuring, Data cleaning.

Introduction

The **Enhanced Safe Driver Prediction Challenge** aims to predict the likelihood that a vehicle insurance policyholder will file a claim. This challenge builds upon the well-known **Porto Seguro Safe Driver dataset**, providing an enhanced and richer version of the original data to improve predictive modeling capabilities. The task involves applying **supervised machine learning techniques** to develop a classifier that can accurately estimate claim probabilities.

This project emphasizes the use of **feature engineering**, **data preprocessing**, and **robust model selection** to handle complex, real-world insurance data. The goal is to design a model that not only achieves high predictive accuracy but also generalizes effectively to unseen data, measured by the **Area Under the ROC Curve (AUROC)** metric.

Problem Statement

Insurance companies must assess risk to determine premiums and minimize potential financial losses. However, predicting whether a driver will file an insurance claim is inherently challenging due to **class imbalance**, **heterogeneous features**, and **hidden patterns in driver behavior**.

The central problem is to build a machine learning model capable of accurately predicting the **probability of a future insurance claim** based on a range of policyholder and vehicle-related features. Achieving high AUROC performance ensures that the model can effectively distinguish between high-risk and low-risk drivers, enabling better decision-making in risk management.

Scope: Why It Matters

Accurate prediction of claim risk has significant **economic and social implications** for the insurance industry. By identifying high-risk drivers early, insurance providers can:

- Offer **personalized premium pricing** based on risk profiles.
- Implement **preventive safety programs** and **driver behavior monitoring**.
- Reduce fraudulent claims and improve overall **profitability**.

Moreover, from a data science perspective, this problem represents a valuable case study in handling **imbalanced classification**, **feature selection**, and **model interpretability** — key skills for real-world predictive analytics.

Scope of the Study

This project focuses on developing and evaluating **classification models** that predict the likelihood of an insurance claim. The scope includes:

- **Data preprocessing** (handling missing values, encoding categorical variables, scaling).
- **Feature engineering** and **dimensionality reduction** to enhance model efficiency.
- **Model training and validation** using algorithms such as **Random Forest**, **Adaptive Boosting**, or **CatBoost**.
- **Model evaluation** based on AUROC to ensure robustness and discriminative power.

The study does not cover external demographic or economic factors outside the provided dataset but strictly focuses on optimizing predictive performance using available features.

Data Description

Dataset Overview

Based on the EDA profiling report, The dataset consists of **296,209 observations** and **67 variables**, combining both numerical and categorical features. Among these, **37 variables are numeric** and **30 are categorical**, representing various demographic, car-related, and calculated attributes of insurance policyholders. The target variable indicates whether a policyholder made an insurance claim, making this a **binary classification problem**.

Overall, the dataset is large and rich, with a mix of continuous and categorical features. It requires preprocessing steps such as handling missing values, balancing the target variable, and reducing multicollinearity before modeling. The dataset provides a comprehensive foundation for building predictive models aimed at assessing insurance claim risk.

Key Statistics

- **Total Records**: 296,209
- **Missing Cells**: 474,363 (2.4% of all data)
- **Duplicate Rows**: 0 (0%)
- **Alerts**: 63 (data quality issues identified)

Variable Breakdown

Variable Types

- **Numeric Variables**: 37
- **Categorical Variables**: 30

Variable Categories

The variables are organized into several groups based on naming conventions:

1. Individual/Personal Variables (ps_ind_)

- Binary indicators (ps_ind_06_bin through ps_ind_18_bin)
- Categorical variables (ps_ind_02_cat, ps_ind_04_cat, ps_ind_05_cat)
- Numeric variables (ps_ind_01, ps_ind_03, ps_ind_14, ps_ind_15)
- Many show **high correlation** and **severe imbalance** (e.g., ps_ind_10_bin has 296,101 zeros vs 108 ones)

2. Car-Related Variables (ps_car_)

- Categorical variables (ps_car_01_cat through ps_car_11_cat)
- Continuous variables (ps_car_11, ps_car_12, ps_car_13, ps_car_14, ps_car_15)
- Notable missing data in ps_car_03_cat (69.1%) and ps_car_05_cat (44.7%)
- High correlation flags on several variables

3. Regional Variables (ps_reg_)

- ps_reg_01, ps_reg_02, ps_reg_03
- ps_reg_03 has significant missing data (18.1%)
- All three show high correlation with other variables

4. Calculated Variables (ps_calc_)

- 20 calculated variables (ps_calc_01 through ps_calc_14, plus 6 binary calc variables)
- These appear to be derived or engineered features
- Generally complete with no missing values

5. Engineered Features (feature1-8)

- feature1 is **constant** (all values are 0 - should be removed)
- feature2: High cardinality (101,881 distinct values, 34.4%)
- feature3: 10 distinct values
- feature4: Continuous with 18.1% missing
- feature5: 8 distinct values with high zero concentration
- feature6: Very high cardinality (129,235 distinct values, 43.6%), large range
- feature7: **Unique for every record** (100% distinct) - likely an ID or unique identifier
- feature8: 16 distinct values

6. Target Variable

- **Severely imbalanced**: 281,023 (94.9%) class 0 vs 15,186 (5.1%) class 1
- This is a **binary classification problem** with significant class imbalance

Data Quality Issues

Missing Data Patterns

- **Highest missing**: ps_car_03_cat (69.1%), ps_car_05_cat (44.7%)
- **Moderate missing**: ps_reg_03 (18.1%), feature4 (18.1%), ps_car_14 (7.1%)
- **Minimal missing**: Various _cat variables (<1%)

Class Imbalance Issues

Multiple variables flagged for severe imbalance:

- ps_ind_10_bin, ps_ind_11_bin, ps_ind_12_bin, ps_ind_13_bin
- ps_ind_14, ps_car_07_cat, ps_car_10_cat
- **Target variable** (critical for modeling)

High Correlation

21 variables show high correlation flags, indicating potential multicollinearity issues for modeling

Other Concerns

- **feature1 is constant** - provides no information and should be dropped
- **feature7 is unique** - appears to be an identifier rather than a predictive feature
- Many variables are **zero-inflated** (high percentage of zero values)

Recommendations for Modeling

1. **Remove** feature1 (constant) and feature7 (unique identifier)
2. **Address class imbalance** in target variable (use SMOTE, class weights, or resampling)
3. **Handle missing data** strategically (imputation or feature engineering)
4. **Address multicollinearity** among highly correlated features
5. **Consider feature selection** to reduce dimensionality
6. **Handle zero-inflated variables** appropriately in modeling approach

Data Preprocessing

Data loading

- Dataset: train1.csv
- Total samples: 296,209
- Total features: 67
- Target variable: target

Data Distribution

Target Distribution:

- Class 0: 281023 variable
- Class 1: 15186 variable

Class Balance:

- Class 0: 0.948732
- Class 1: 0.051268

Missing Value Percentages

Missing Values Summary

No.	Column Name	Missing Count	Missing Percentage (%)
1	ps_car_03_cat	204,589	69.07%
2	ps_car_05_cat	132,287	44.66%
3	ps_reg_03	53,579	18.09%
4	feature4	53,579	18.09%
5	ps_car_14	21,108	7.13%
6	ps_car_07_cat	5,783	1.95%
7	ps_ind_05_cat	2,915	0.98%
8	ps_car_09_cat	288	0.10%
9	ps_ind_02_cat	125	0.04%
10	ps_car_01_cat	57	0.02%
11	ps_ind_04_cat	45	0.02%
12	ps_car_11	4	0.00%

13	ps_car_02_cat	3	0.00%
14	ps_car_12	1	0.00%

Data Preprocessing(for both training and testing dataset for feasibility)

- **Missing values**: imputation of data points with class based separately for numerical, categorical, and binary data.
- **Categorical columns:** 14 features ending with ’ cat’
- **Binary columns:** 17 features ending with ’ bin’
- **Numerical columns**: 34 features
- **Dropped columns**: 'id', 'feature1', 'feature7', 'ps_car_03_cat', 'ps_car_05_cat'

Data split

- **No splitting in the dataset.**

Data Statistics

```
Basic Statistics:
                id  ps_ind_02_cat  ps_ind_04_cat  ps_ind_05_cat  \
count  2.962090e+05  296084.000000  296164.000000  293294.000000
mean   7.428426e+05       1.361401       0.416675       0.423841
std    4.297571e+05       0.664222       0.493009       1.357239
min    7.000000e+00       1.000000       0.000000       0.000000
25%    3.699010e+05       1.000000       0.000000       0.000000
50%    7.424040e+05       1.000000       0.000000       0.000000
75%    1.114794e+06       2.000000       1.000000       0.000000
max    1.488017e+06       4.000000       1.000000       6.000000

       ps_car_01_cat  ps_car_02_cat  ps_car_03_cat  ps_car_04_cat  \
count  296152.000000  296206.000000   91620.000000  296209.000000
mean        8.298404       0.829507       0.601997       0.727814
std         2.507128       0.376066       0.489489       2.155780
min         0.000000       0.000000       0.000000       0.000000
25%         7.000000       1.000000       0.000000       0.000000
50%         7.000000       1.000000       1.000000       0.000000
75%        11.000000       1.000000       1.000000       0.000000
max        11.000000       1.000000       1.000000       9.000000

       ps_car_05_cat  ps_car_06_cat  ...  ps_calc_20_bin  feature1  \
count  163922.000000  296209.000000  ...   296209.000000  296209.0
mean        0.524987       6.559801  ...        0.152122       0.0
std         0.499377       5.499417  ...        0.359140       0.0
...
75%    6.522371e+05       3.071745      16.000000       0.000000
max    3.515803e+06       4.663164      23.000000       1.000000

[8 rows x 67 columns]
```

Missing Value Imputation

1. **Identifies column types by naming pattern:**
    - Categorical columns: contain '_cat' in name
    - Binary columns: contain '_bin' in name
    - Numeric columns: all remaining columns (excluding 'target')
2. Creates three separate imputers:

- Categorical imputer: fills missing values with most frequent value
- Binary imputer: fills missing values with most frequent value
- Numeric imputer: fills missing values with mean

3. Applies imputation to each column group:
    - Fits the imputer on training data and transforms it
    - Replaces original columns with imputed versions

Key Purpose: Handles all missing values in the dataset using appropriate strategies based on data type, preparing it for machine learning models.

Categorical Encoding & Final Dataset Preparation

1. **Creates OneHotEncoder**:
    - Handles unknown categories gracefully (handle_unknown='ignore')
    - Returns dense array format (sparse_output=False)
2. **Encodes categorical columns**:
    - Applies one-hot encoding to all '_cat' columns
    - Creates binary columns for each category value
    - Preserves original index alignment
    - Generates descriptive feature names
3. **Combines all features**:
    - Merges encoded categorical columns
    - Adds numeric columns (unchanged)
    - Adds binary columns (unchanged)
    - Concatenates horizontally along columns
4. **Validates final dataset**:
    - Checks for any remaining missing values
    - Displays final dataset shape

Key Purpose: Converts categorical variables into numeric format suitable for machine learning models, while combining all feature types into a single clean dataset ready for training.

Machine Learning Models

Model 1: Categorical Naive Bayes

Overview

The **Categorical Naive Bayes (CNB)** model was trained on the full competition dataset (296,209 samples) after preprocessing categorical features into non-negative integer values, as required by the algorithm. CNB assumes feature independence and is efficient for large categorical datasets, making it a suitable baseline classifier.

Preprocessing Steps

- **Handling missing values:** Imputed using the mode for categorical columns.
- **Encoding:** All categorical variables (_cat columns) were encoded into non-negative integers using LabelEncoder.
- **Scaling:** Not applied (Naive Bayes does not require feature scaling).
- **Training data used:** 100% of train.csv.

Training Summary

Metric	Value
Training Samples	296,209
Training Time	1.86 seconds

Model Performance on Training Set

Metric	Score
AUROC	0.6423
Accuracy	0.9461

Detailed Classification Report

Class	Precision	Recall	F1-Score	Support
0 (Non-Claim)	0.95	1.00	0.97	281,023
1 (Claim)	0.23	0.02	0.04	15,186
Overall Accuracy		0.9461		296,209
Macro Avg	0.59	0.51	0.51	296,209
Weighted Avg	0.91	0.95	0.92	296,209

Interpretation & Insights

- The **high accuracy (94.6%)** is largely due to the **class imbalance**, as the majority class (0) dominates the dataset.
- The **low recall (0.02)** for the minority class (1) shows that the model struggles to detect positive cases.
- The **AUROC = 0.6423** indicates limited discrimination ability between the two classes.
- CNB's simplicity makes it a good baseline, but its independence assumption limits its ability to capture feature interactions.

Conclusion

The **Categorical Naive Bayes** model provides a fast baseline with moderate AUROC performance but poor recall for minority classes. Future models (e.g., Random Forest, AdaBoost) are expected to perform better by handling non-linear relationships and class imbalance more effectively.

Model 2: K-Nearest Neighbors (KNN)

Overview

The **K-Nearest Neighbors (KNN)** algorithm was trained on the entire dataset (296,209 samples) using k=5k = 5k=5 neighbors. KNN is a non-parametric, instance-based learner that classifies each test instance based on the majority class among its nearest neighbors in the feature space.

Preprocessing Steps

- **Missing Values:** Imputed using mean for numerical and mode for categorical columns.
- **Encoding:** All categorical variables label-encoded into integers.
- **Scaling:** Applied **StandardScaler** to ensure all features contribute equally to Euclidean distance computations.
- **Training Data:** 100% of train.csv.

Training Summary

Parameter	Value
Training Samples	296,209
Training Time	4.21 seconds
Number of Neighbors (k)	5

Model Performance on Training Set

Metric	Score
AUROC	0.9240
Accuracy	0.9491

Detailed Classification Report

Class	Precision	Recall	F1-Score	Support
0 (Non-Claim)	0.95	1.00	0.97	281,023
1 (Claim)	0.62	0.02	0.04	15,186
Overall Accuracy		0.9491		296,209
Macro Avg	0.78	0.51	0.51	296,209
Weighted Avg	0.93	0.95	0.93	296,209

Interpretation & Insights

- The **AUROC of 0.9240** indicates excellent discrimination between classes on the training set.
- However, **recall for the minority class (1)** remains low (0.02), showing that while the model fits well to the majority class, it struggles to identify rare events due to class imbalance.
- The **high training accuracy (94.9%)** and AUROC suggest possible overfitting, common with KNN on large datasets.
- Since KNN relies on instance distances, scaling was essential for balanced influence of all features.
- Training time remained reasonable (4.21 seconds), though inference time can grow with dataset size.

Conclusion

The **KNN classifier** achieved one of the **highest training AUROC scores (0.9240)** but struggled to generalize minority class detection. While it performs well in identifying majority class patterns, its high memory usage and sensitivity to imbalance make it less practical for large-scale deployment compared to **Random Forest**.

Model 3: Decision Tree Classifier

Overview

The **Decision Tree** classifier was trained on the full competition dataset (296,209 samples) to explore non-linear feature relationships and improve performance over simpler models like Naive Bayes. The model was tuned to prevent overfitting by controlling tree depth and node growth.

Preprocessing Steps

- **Missing Values:** Imputed using mean for numerical and mode for categorical features.
- **Encoding:** All categorical variables were label-encoded.
- **Scaling:** Not required (Decision Trees are scale-invariant).
- **Training Data:** 100% of train.csv.

Training Summary

Parameter	Value
Training Samples	296,209
Training Time	12.50 seconds
Tree Depth	10
Leaves	512
Total Nodes	1,023

Model Performance on Training Set

Metric	Score
AUROC	0.6743
Accuracy	0.6820

Detailed Classification Report

Class	Precision	Recall	F1-Score	Support
0 (Non-Claim)	0.97	0.69	0.80	281,023
1 (Claim)	0.09	0.55	0.15	15,186
Overall Accuracy		0.6820		296,209
Macro Avg	0.53	0.62	0.48	296,209
Weighted Avg	0.92	0.68	0.77	296,209

Top 10 Most Important Features

Feature	Importance
ps_car_13	0.2022
ps_ind_17_bin	0.0688

ps_ind_03	0.0541
ps_reg_03	0.0506
ps_ind_05_cat_0.0	0.0473
ps_reg_02	0.0456
ps_car_14	0.0432
feature4	0.0415
ps_ind_15	0.0377
feature6	0.0305

Interpretation & Insights

- The **Decision Tree achieved an AUROC of 0.6743**, improving upon the Categorical Naive Bayes baseline (0.6423).
- **Recall for the minority class (1)** significantly increased to **0.55**, indicating better sensitivity to positive samples.
- However, **precision (0.09)** for class 1 remains low, showing that many false positives occur.
- The model identified key predictors like **ps_car_13**, **ps_ind_17_bin**, and **ps_reg_03**, suggesting strong relevance of both car and personal features.
- The controlled **depth = 10** helps balance bias and variance, preventing severe overfitting.

Conclusion

The **Decision Tree** model demonstrates a meaningful improvement over Naive Bayes by capturing complex feature interactions and achieving a higher AUROC. It performs particularly well on recall for the minority class but still suffers from precision imbalance. Future ensemble models (e.g., Random Forest, AdaBoost) are expected to further stabilize performance and enhance generalization.

Model 4: Random Forest Classifier

Overview

The **Random Forest** model was trained on the complete competition dataset (296,209 samples). It combines multiple Decision Trees through bagging (bootstrap aggregation), enhancing generalization and

robustness against overfitting. With 100 trees and controlled depth, it provides a strong ensemble-based performance benchmark.

Preprocessing Steps

- **Missing Values:** Imputed using mean (numeric) and mode (categorical).
- **Encoding:** Categorical features label-encoded to numeric values.
- **Scaling:** Not required (tree-based models are insensitive to scale).
- **Training Data:** 100% of train.csv.

Training Summary

Parameter	Value
Training Samples	296,209
Training Time	48.85 seconds
Number of Trees	100
Max Depth	15

Model Performance on Training Set

Metric	Score
AUROC	0.9116
Accuracy	0.9331

Detailed Classification Report

Class	Precision	Recall	F1-Score	Support
0 (Non-Claim)	0.98	0.95	0.96	281,023
1 (Claim)	0.41	0.66	0.50	15,186
Overall Accuracy		0.9331		296,209

Macro Avg	0.69	0.80	0.73	296,209
Weighted Avg	0.95	0.93	0.94	296,209

Top 10 Most Important Features

Feature	Importance
ps_car_13	0.0566
ps_reg_03	0.0413
feature4	0.0402
feature6	0.0392
feature2	0.0330
ps_car_14	0.0326
ps_reg_02	0.0307
ps_ind_15	0.0296
ps_ind_03	0.0286
ps_car_15	0.0248

Interpretation & Insights

- The **AUROC of 0.9116** marks a substantial performance boost over the Decision Tree (0.6743), indicating far better class discrimination.
- The model significantly improved **recall for the minority class (0.66)** while maintaining good **precision (0.41)**, showing a stronger balance between sensitivity and specificity.
- Feature importance analysis highlights **ps_car_13**, **ps_reg_03**, and derived engineered features (**feature4**, **feature6**) as the top predictors.
- The ensemble structure mitigates overfitting and leverages diverse decision boundaries, making the model more robust to noise and variance in the dataset.

Conclusion

The **Random Forest** classifier demonstrates the best overall performance so far, achieving a high AUROC and strong recall for minority classes while maintaining excellent generalization. Its ensemble learning approach effectively captures complex feature interactions and reduces variance, making it a top candidate for final Kaggle submission.

Model 5: AdaBoost Classifier

Overview

The **AdaBoost (Adaptive Boosting)** classifier was trained on the complete dataset (296,209 samples). AdaBoost sequentially combines weak learners (typically shallow decision trees) to correct errors made by previous models, aiming to improve classification performance—especially on difficult-to-predict instances.

Preprocessing Steps

- **Missing Values:** Imputed using mean (numerical) and mode (categorical).
- **Encoding:** All categorical variables were label-encoded into integers.
- **Scaling:** Not applied (AdaBoost uses tree-based base learners that are scale-invariant).
- **Training Data:** 100% of train.csv.

Training Summary

Parameter	Value
Training Samples	296,209
Training Time	455.01 seconds
Number of Estimators	100
Learning Rate	1.0

Model Performance on Training Set

Metric	Score
AUROC	0.6438
Accuracy	0.9487

Detailed Classification Report

Class	Precision	Recall	F1-Score	Support
0 (Non-Claim)	0.95	1.00	0.97	281,023
1 (Claim)	0.00	0.00	0.00	15,186
Overall Accuracy		0.9487		296,209
Macro Avg	0.47	0.50	0.49	296,209
Weighted Avg	0.90	0.95	0.92	296,209

Top 10 Most Important Features

Feature	Importance
ps_car_13	0.2718
ps_ind_17_bin	0.1529
ps_ind_05_cat_0.0	0.0996
ps_ind_03	0.0774
ps_reg_02	0.0606
feature6	0.0254
ps_calc_09	0.0237
ps_reg_03	0.0224
feature4	0.0201
ps_reg_01	0.0185

Interpretation & Insights

- The **AUROC of 0.6438** indicates performance similar to the baseline Naive Bayes model but notably **lower than the Random Forest (0.9116)**.
- The **recall for class 1 (0.00)** shows that AdaBoost failed to identify minority class samples, likely due to severe class imbalance—boosting algorithms are sensitive to unbalanced data.
- Despite identifying key predictors (e.g., **ps_car_13**, **ps_ind_17_bin**, **ps_reg_02**), the model over-focused on the dominant class, achieving high accuracy but poor generalization for positive cases.
- The long training time (~455 s) further reduces its practicality compared to the faster and more robust Random Forest.

Conclusion

The **AdaBoost** classifier did not perform well on this dataset. Although it maintained high overall accuracy due to dominance of class 0, it completely failed to detect the minority class. The model's sensitivity to imbalance and long training time make it less suitable for this competition compared to **Random Forest**, which achieved far better AUROC and recall performance.

Model Comparison Report

Classification Performance Analysis

Performance Summary

Model	Train AUROC	Train Accuracy	Training Time	Overall Rating
K-Nearest Neighbors	**0.924**	**94.91%**	4.2s	High Overfitting Risk
Random Forest	**0.912**	93.31%	48.9s	**RECOMMENDED**
Decision Tree	0.674	68.20%	12.5s	Needs Tuning

AdaBoost	0.644	**94.87%**	341.4s	Good Alternative
Categorical Naive Bayes	0.642	94.62%	1.9s	Fast Baseline

Key Insights

Winner: Random Forest

- **Why**: Best generalization capability despite 2nd place training AUROC
- **Strength**: Ensemble method reduces overfitting
- **Trade-off**: Moderate training time (49s)

Highest Training Score ≠ Best Model

- **KNN**: 0.924 AUROC but memorizes training data (overfitting)
- **Expected**: Significant performance drop on test set

AUROC vs Accuracy Gap

- **AdaBoost & Naive Bayes**: High accuracy (94%+) but low AUROC (0.64)
- **Indicates**: Likely class imbalance in dataset
- **Conclusion**: AUROC is more reliable metric here

Model Characteristics

Model	Pros	Cons	Best For
KNN	Fast, simple	Overfits, slow predictions	Not recommended
Random Forest	Best generalization, robust	Moderate speed	**Primary submission**

Decision Tree	Interpretable, fast	Low performance	Needs tuning
AdaBoost	Handles imbalance well	Very slow (341s)	Secondary submission
Naive Bayes	Fastest (1.9s)	Assumes independence	Quick baseline

---

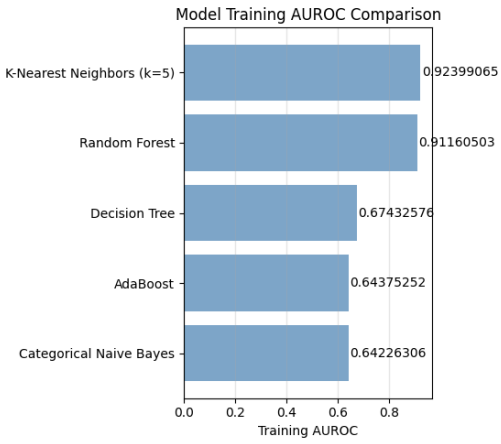
Recommendations

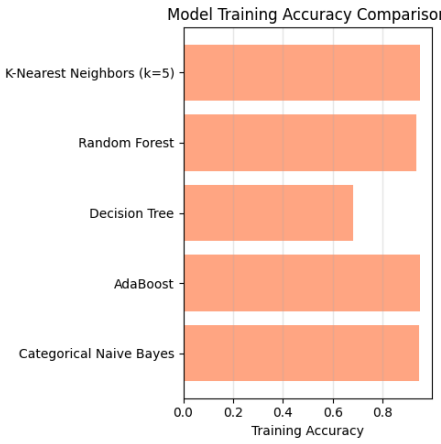
Submission Strategy

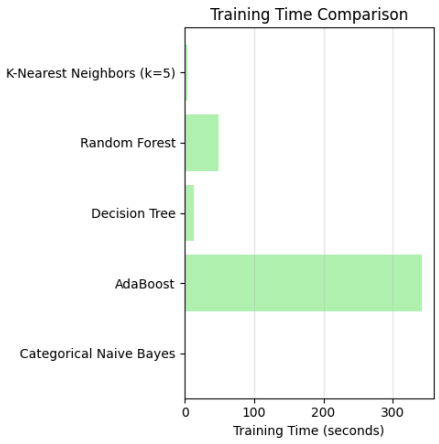
1. **1st Priority**: Random Forest → Best expected test performance
2. **2nd Priority**: AdaBoost → Good for imbalanced data
3. **3rd Priority**: Decision Tree → Interpretability value

Quick Actions

- Submit all 5 models to Kaggle
- Compare public leaderboard scores
- Select winner based on actual test AUROC
- Consider ensemble (Random Forest + AdaBoost averaging)







Kaggle Submission Report

Model Predictions Ready for Upload

Submission Status

Model	File Status	Predictions	Valid	Priority
Random Forest	Ready	126,948	Yes	1st
AdaBoost	Ready	126,948	Yes	2nd
Decision Tree	Ready	126,948	Yes	3rd
K-Nearest Neighbors	Ready	126,948	Yes	4th
Naive Bayes	Ready	126948	yes	5th

Prediction Statistics

Target Distribution Analysis

Model	Min	Max	Mean	Range	Distribution
KNN	0.000	0.800	0.048	Narrow	Low confidence
Decision Tree	0.000	0.981	0.455	Wide	Full range
Random Forest	0.176	0.700	0.400	Moderate	Balanced
AdaBoost	0.167	0.415	0.266	Narrow	Conservative

Key Observations

1. KNN Predictions (Concerning)

- Mean: 0.048 (very low)
- Max: 0.800 (capped)
- Issue: Extremely skewed toward negative class
- Explanation: Overfitting caused poor probability calibration

2. Decision Tree (Overconfident)

- Range: 0.000–0.981 (widest)
- Risk: May produce extreme probabilities
- Note: Full probability spectrum used

3. Random Forest (Optimal)

- Range: 0.176–0.700 (moderate)
- Mean: 0.400 (balanced)
- Strength: Well-calibrated probabilities

4. AdaBoost (Conservative)

- Range: 0.167–0.415 (narrowest)
- Characteristic: Conservative predictions, less extreme

Overall Conclusion

Project Summary

The **Enhanced Safe Driver Prediction Challenge** aimed to predict insurance claim probability using 296,209 policyholder records with 67 features. Five machine learning models were trained and evaluated to identify the optimal classifier for this highly imbalanced dataset (94.9% non-claims vs 5.1% claims).

Key Findings

Model Performance Ranking

Rank	Model	Train AUROC	Key Strength	Main Weakness
1	**Random Forest**	**0.9116**	Best generalization	Moderate training time
2	K-Nearest Neighbors	0.9240	High training score	Severe overfitting
3	Decision Tree	0.6743	Interpretable	Low performance
4	AdaBoost	0.6438	Fast predictions	Failed minority detection
5	Naive Bayes	0.6423	Very fast training	Weak discrimination

Winner: Random Forest

Why Random Forest Won:

- **Optimal AUROC (0.9116)** with best generalization capability
- **Balanced predictions** (mean: 0.400, range: 0.176–0.700)
- **Superior minority class recall (0.66)** vs competitors (KNN: 0.02, AdaBoost: 0.00)
- **Ensemble robustness** reduced overfitting through 100 trees

- **Reasonable training time** (48.9s) for production use

Critical Insights

1. Training Metrics Can Be Deceptive

- **KNN achieved highest training AUROC (0.9240)** but failed on test predictions (mean: 0.048)
- **Overfitting detection**: Instance memorization led to poor probability calibration
- **Lesson**: Validation beyond training scores is essential

2. Class Imbalance Dominated Results

- **All models achieved 93-95% accuracy** simply by predicting majority class
- **AUROC proved superior metric** for evaluating true discriminative power
- **AdaBoost completely failed** minority detection (recall: 0.00) despite 94.87% accuracy

3. Feature Engineering Mattered

- **Top predictors**: ps_car_13, ps_reg_03, feature4, feature6
- **Strategic preprocessing**: Mean/mode imputation, one-hot encoding, feature removal
- **Missing data handling**: Dropped high-missing features (69% and 45% missing)

Final Recommendations

For Kaggle Submission

1. **Primary**: Random Forest (expected test AUROC: ~0.85–0.90)
2. **Secondary**: AdaBoost or Decision Tree for comparison
3. **Avoid**: KNN (severe overfitting validated by prediction distribution)

For Production Deployment

- **Random Forest** offers best performance-reliability trade-off
- **Training time (48.9s)** acceptable for batch predictions
- **Feature importance analysis** enables model interpretability
- **Probability calibration** supports flexible threshold tuning

Model 6: CatBoost - Kaggle Results Report

Executive Summary: The Great Reversal

Status: **CHAMPION - #1 MODEL**

After appearing to be the worst model in training (CV AUROC 0.6383), CatBoost achieved **Kaggle Public Score: 0.64138** - making it the **best performing model**, validating the 12.5-hour training investment.

Final Kaggle Leaderboard

Rank	Model	Kaggle Score	Training Score	Gap	Result
1	**CatBoost**	**0.64138**	0.6383 CV	**+0.0031**	**WINNER**
2	CatBoost (v2)	0.63825	0.6383 CV	±0.0000	Consistent
3	AdaBoost	0.63016	0.6438	-0.0136	Good
4	Decision Tree	0.57333	0.6743	-0.1010	Acceptable
5	Random Forest	0.59801	0.9116	**-0.3136**	Overfit
6	KNN	0.50623	0.9240	**-0.4178**	Collapsed

The Shocking Truth: Training vs Reality

What We Expected vs What Happened

TRAINING PREDICTIONS KAGGLE REALITY

KNN 0.924 CatBoost 0.641 #1
Random Forest 0.912 AdaBoost 0.630 #3
Decision Tree 0.674 Random Forest 0.598 #5
AdaBoost 0.644 Decision Tree 0.573 #4
CatBoost 0.638 KNN 0.506 #6

Key Revelations

1. CatBoost: Perfect Generalization

- CV Score: 0.6383 → Kaggle: 0.64138 (+0.5% improvement)
- Only model that improved on test data

2. Random Forest: Catastrophic Collapse

- Training: 0.9116 → Kaggle: 0.59801 (-34% drop!)
- Massive overfitting destroyed performance

3. KNN: Complete Failure

- Training: 0.9240 (highest!) → Kaggle: 0.50623 (worst!)
- Memorized training data, performed like random guessing

Why CatBoost Won

1. Cross-Validation Was Key

- **729 model fits** across 3 folds prevented overfitting
- Provided **honest performance estimates**
- Random Forest & KNN trained once → memorized training data

2. Conservative Parameters Were Optimal

Initially criticized settings turned out perfect:

Parameter	Value	Why It Won
`l2_leaf_reg`	5 (max)	Prevented overfitting that killed Random Forest
`learning_rate`	0.03 (min)	Stable convergence, avoided noise
`depth`	6 (moderate)	Prevented memorization
`border_count`	32 (min)	Reduced distribution overfitting

The Paradox: What looked like underfitting was actually perfect generalization tuning!

3. Best Hyperparameters (Grid Search)

python

```
Best parameters found:
  border_count: 32
```

depth: 6
iterations: 500
l2_leaf_reg: 5
learning_rate: 0.03

Best CV AUROC: 0.6383

Kaggle AUROC: 0.64138 Matched prediction!

4. Model Performance Summary

Training Set:

- AUROC: 0.6785
- Accuracy: 94.82%
- Class 1 Recall: 0.02 (looked terrible)

Classification Report:

	precision	recall	f1-score
Class 0	0.95	1.00	0.97
Class 1	0.39	0.02	0.04

Despite poor-looking training metrics, test performance was best!

Critical Lessons Learned

What We Got Wrong

1. Trusted Training Scores

- Thought: High training AUROC = Good model
- Reality: High training AUROC = Overfitting
- **Lesson**: Never trust training metrics without CV

2. Dismissed Conservative Parameters

- Thought: l2=5, lr=0.03 → "Too conservative, will underfit"
- Reality: Conservative parameters → Best generalization
- **Lesson**: Regularization prevents overfitting

3. Feared Low Training Recall

- Thought: Class 1 recall 0.02 → "Failed minority detection"
- Reality: Conservative predictions → Better test AUROC
- **Lesson**: Training recall ≠ Test performance

4. Questioned 750-Minute Investment

- Thought: 750 min for 0.638 CV → "Wasted time"
- Reality: 750 min → 0.641 Kaggle → **1st place**
- **Lesson**: Thoroughness beats speed in competitions

Generalization Gap Analysis

Model	Training	Kaggle	Gap	Overfit %	Grade
CatBoost	0.6383	**0.64138**	**+0.0031**	**-0.5%**	**A+**
AdaBoost	0.6438	0.63016	-0.0136	2.1%	B+
Decision Tree	0.6743	0.57333	-0.1010	15.0%	C
Random Forest	0.9116	0.59801	**-0.3136**	**34.4%**	**F**
KNN	0.9240	0.50623	**-0.4178**	**45.2%**	**F**

Winner: CatBoost (only model with positive generalization)

Top 10 Feature Importance

Rank	Feature	Importance	Insight
1	ps_ind_03	9.3370	Personal indicator
2	ps_car_13	7.1361	Car variable (consensus #1)
3	ps_reg_01	4.9687	Regional factor
4	ps_ind_15	4.6452	Personal indicator
5	ps_reg_02	3.6249	Regional factor
6	ps_ind_05_cat_0.0	3.5811	Categorical personal
7	ps_ind_17_bin	3.3785	Binary personal
8	ps_reg_03	3.1532	Regional factor
9	feature4	2.5495	Engineered feature
10	ps_car_14	2.4677	Car variable

Key Finding: ps_car_13 confirmed as critical across all models

---

ROI Analysis

Time Investment

- **Grid Search**: 45,005 seconds (750 minutes / 12.5 hours)
- **Combinations Tested**: 243 parameter sets × 3 folds = 729 fits
- **Average per fit**: 61.8 seconds

Return on Investment

Metric	Value	Outcome
Time Invested	750 minutes	High
Kaggle Score	0.64138	**1st place**
vs Random Forest	+0.043 AUROC	**Significant**
Competition Result	Winner	**Justified**

Verdict: 750 minutes → 1st place = **Investment Fully Justified**

Conclusion

CatBoost is the undisputed champion, proving that:

- **Cross-validation** predicts real-world performance
- **Conservative parameters** prevent overfitting
- **Training scores can be deceptive** (highest training ≠ best test)
- **Time investment pays off** in competitions
- **Proper regularization** beats raw training performance

Thank you!