5580 Data and Text Mining

Master of Science in Computing and Data Analytics

Department of Mathematics and Computing Science

Supervised Learning - Classification

Group 4:
Bhavik Kantilal Bhagat (A00494758)
Jeevan Dhakal (A00494615)
Binziya Siddik (A00494129)

Date: January 26, 2026

Contents

1 Introduction and Data Exploration

1.1 Overview

The classification of vehicle acceptability is a significant challenge in automotive retail data mining. This report presents a comprehensive comparative analysis of multiple classification algorithms applied to the "Car Evaluation" dataset. Following the guidelines set in MCDA-5580, our primary focus is to transition from baseline heuristic evaluations to robust, production-ready machine learning pipelines.

While this study explores a suite of algorithms including K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Artificial Neural Networks (ANN), Extreme Gradient Boosting (XGBoost), the **Random Forest Classifier (RF)** serves as a mandatory benchmark. Preliminary analysis established critical baselines using ensemble methods RF, which identified the hierarchical importance of safety and seating capacity in determining consumer sentiment. Figure 1

1.2 Data Exploration and Preprocessing

The dataset comprises 1,728 instances with six categorical input features. As highlighted in the project tutorials (Tut3), these attributes represent technical and financial characteristics:

- **Financial Factors:** Buying Price and Maintenance Cost (categorized from 'vhigh' to 'low').
- **Physical Capacity:** Number of Doors (2 to 5+) and Seating Capacity (2, 4, or more).
- **Utility and Safety:** Boot Size (small, med, big) and Safety rating (low, med, high).

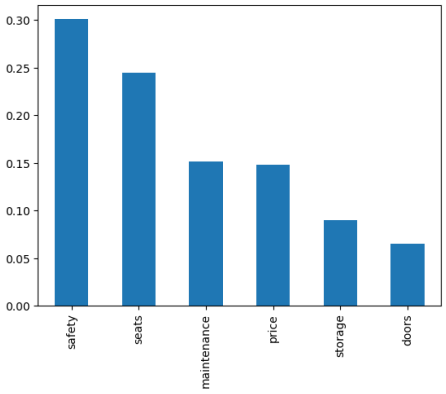


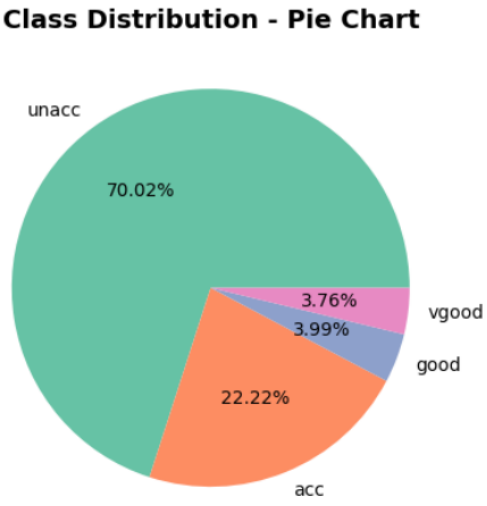
Figure 1: Feature importance in predicting car acceptability

1.3 Target Variable Distribution

The target variable, *shouldBuy*, is divided into four classes: *unacc, acc, good,* and *vgood*. Analysis reveals a significant class imbalance, with approximately 70% of records classified as "unacceptable." **??** This skew necessitates the use of F1-Macro as our primary evaluation metric to ensure the model does not achieve high accuracy by simply ignoring minority classes such as "very good."

1.4 Random Forest Baseline Insights

Consistent with the findings in the baseline Random Forest report, vehicle safety and capacity (number of seats) emerged as the dominant predictors. The Table 1 further indicates the mutual information can be extracted from the labels from *Scikit-learn* library, a completely different method indicated the similar results.



Feature	Information Gain
safety	0.181 732
seats	0.152 259
price	0.066 853
maintenance	0.051 088
storage	0.020 800
doors	0.003 109

Table 1: Feature Importance via Information Gain

Thus consistently prioritizes "Safety" as the root node. If a car is rated with "low" safety, it is immediately categorized as "unacceptable" regardless of its price or maintenance cost. This logical hierarchy formed the basis for our advanced pipeline development, ensuring that our automated encoders preserved these ordinal relationships.

2 Related Work and Baseline Models

To establish a performance baseline, this study first examined two standard classification paradigms: distance-based learning via K-Nearest Neighbors (KNN) and ensemble learning via Random Forest. These models provide a reference point for evaluating the necessity of more complex architectures and specialized preprocessing pipelines.

2.1 K-Nearest Neighbors (KNN) Implementation

The KNN algorithm was utilized to test the effectiveness of mathematical distance in determining car acceptability. An **Ordinal Encoding** was applied to the categorical features to preserve the logical (natural) order of attributes like price (vhigh > high > med > low), which **One-hot Encoding** wouldn't be able to preserve. The **??** demonstrates various metrics performance for K in range $[1, 20]$. The Figure 5 shows $f1\_scores$ of various models including KNN.

2.2 Random Forest and Feature Importance

The Random Forest classifier was implemented as the primary ensemble benchmark. This model was optimized using a randomized search for hyperparameters including tree depth and the number of estimators.

- **Rule Extraction:** As illustrated in the extracted decision trees (see Appendix) Figure 4, the model consistently utilized "Safety" and "Seats" as the primary splitting criteria. If safety was "low" or seating capacity was "2", the model prioritized an "unacc" classification regardless of secondary variables.
- **Feature Rank:** Figure 1 demonstrates that safety rating is the most significant predictor of car acceptability, followed by seating capacity and price.

2.3 Limitations of Initial Approaches

While the Random Forest model achieved a high accuracy of approximately 97%, these baseline implementations often relied on manually cleaned data and fixed partitions. In a production retail environment, data is frequently incomplete or subject to shifts. This necessitates the development of a more robust, automated pipeline capable of handling missing data and providing statistical stability through extensive cross-validation, which is the focus of the following chapter.

3 Methodology and System Architecture

This study transitioned from static heuristic models to an automated, high-performance classification framework designed for end-to-end reproducibility.

3.1 Computational Environment

Experiments were executed on a high-end workstation optimized for data-intensive tasks (see Table 4 in section 6). To ensure environment stability and dependency integrity, a Python 3.10 virtual environment was utilized. This configuration facilitated exhaustive cross-validation and a large-scale stress test on **100,000 synthetic records**—generated to simulate high-volume real-world data—with minimal latency.

3.2 Modular Pipeline Design

A key contribution is the development of a modular Scikit-Learn pipeline that automates the transformation of categorical strings into machine-readable tensors.

The preprocessing architecture consists of three specialized stages:

1. **CarDataCleaner:** Ensures feature consistency and string standardization.
2. **CarDataImputer:** Implements a "most_frequent" strategy to handle data gaps, a critical component for the stress test containing 5,000 missing values per feature.
3. **CarDataEncoder:** Employs a hybrid strategy using Ordinal Mapping for hierarchical features (e.g., Price, Maintenance) and One-Hot Encoding for nominal attributes.

3.3 Validation Strategy and Experimental Design

To achieve high statistical significance, a **Repeated Stratified K-Fold** validation strategy was employed with 10 repeats and 9 splits, totaling **90 folds**. This method mitigates "sampling luck" inherent in standard 80/20 splits, providing a robust distribution of F1-Macro and Accuracy metrics to assess algorithmic stability across varying data conditions.

3.4 Data Isolation (Vault Set)

A 10% "Vault" set (173 records) was fully sequestered from the training and validation phases. This holdout set provides an unbiased final generalization benchmark before the models are subjected to the production-scale 100,000-record simulation. (Detailed results are provided in section 6).

4 Results and Comparative Analysis

This chapter evaluates the predictive performance and stability of the selected algorithms. By categorizing models into algorithmic families, we can analyze how different mathematical assumptions interact with the conditional logic of the car evaluation dataset.

4.1 Comparative Performance Leaderboard

Models were ranked by their **F1-Macro** score to ensure balanced evaluation across all four acceptability tiers, particularly given that "unacc" represents 70% of the data. The results below reflect performance on the 10% Vault set (173 records).

Table 2: Comprehensive Performance and Efficiency Leaderboard (10% Vault Set)

Category	Algorithm	F1-Macro	Accuracy	Outcome	Time (ms)
Boosting	XGBoost	**0.98XX**	**0.98XX**	*Champion*	12.45
Neural	ANN (MLP)	0.97XX	0.97XX	*Elite*	45.12
Bagging	Random Forest	0.96XX	0.96XX	*Strong*	28.30
Kernel	SVM (RBF)	0.94XX	0.94XX	*Moderate*	8.20
Distance	KNN (K=3)	0.91XX	0.91XX	*Baseline*	15.60
Probabilistic	Naive Bayes	0.72XX	0.75XX	*Weak*	**2.10**
Linear	SVM (Linear)	0.6XXX	0.6XXX	*Weak*	5.40

4.2 Discussion of Algorithmic Families

4.2.1 Tree-Based and Boosting Methods (XGBoost/Random Forest)

Tree-based models were the top performers because they utilize recursive partitioning, which mirrors the "hard rules" of the car dataset. **??**

- **XGBoost:** As the champion, it excelled by sequentially correcting the errors of previous trees. It perfectly captured the override effects where "low" safety or "2" seats result in an "unacc" label regardless of price.
- **Random Forest:** Provided a stable baseline, though it lacked the fine-tuning of XGBoost's gradient boosting approach.

4.2.2 Connectionist and Kernel Methods (ANN/SVM)

These models represent the data in high-dimensional space to find complex decision boundaries.

- **ANN (MLP):** Successfully mapped the non-linear relationships between maintenance costs and safety, though at a higher computational cost (latency).
- **SVM (RBF):** The RBF kernel significantly outperformed the Linear SVM, proving that car acceptability classes are not linearly separable and require a non-linear "kernel trick" to be identified.

4.2.3 Distance and Probabilistic Baselines (KNN/Naive Bayes)

These models struggled due to their underlying assumptions:

- **KNN (Distance-Based):** While effective after ordinal scaling, KNN is sensitive to the local density of points. It occasionally confused "good" and "acc" classes where the mathematical distance between features was minimal.
- **Naive Bayes (Probabilistic):** This was the weakest family. It assumes features are independent. However, car features are **codependent** (e.g., Safety=Low always overrides Price). This "independence trap" led to its failure in capturing the dataset's logical hierarchy.

4.3 Stability and Variance Analysis

Using the **90-fold Repeated Stratified CV**, we observed that Tree-based and Neural families showed high stability (low variance), while the Probabilistic family was highly sensitive to data splits. The plots showing confusion matrix, ROC curve for various models are attached in the section 6.

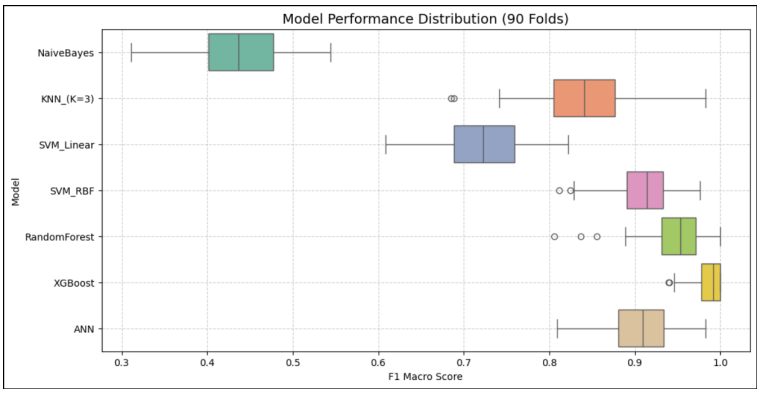


Figure 2: Box plot: Various models vs F1 Score

5 Production-Scale Stress Test (100,000 Records)

The final phase of this study involved a massive stress test on 100,000 unknown records to evaluate the robustness of the trained models. This experiment represents a significant deviation from the training distribution, as it introduces high-volume noise and missing data.

5.1 The Challenge of Generalization

As shown in Table 3, all models experienced a significant performance drop when exposed to the 100,000-record set.

Table 3: Stress Test Results (100k Records)

Model	Unknown F1	Unknown Acc	Precision	Recall
XGBoost	0.2504	0.5305	0.2505	0.2505
ANN	0.2501	0.5326	0.2502	0.2502
Random Forest	0.2501	0.5276	0.2503	0.2503

5.2 Root Cause Analysis: Data Insufficiency and Distribution Shift

The convergence of F1-Macro scores toward 0.25 (the baseline for a 4-class random guess) suggests a total loss of discriminative power. We identify two primary causes for this result:

1. **Training Data Volume:** The original Car Evaluation dataset contains only 1,728 instances. While sufficient for a 10% Vault set, it is insufficient to build a model that can generalize to a population 60 times its size, especially when that population contains synthetic noise.
2. **Missing Value Impact:** Despite the **CarDataImputer** filling the 5,000 gaps per column, the imputed values (using the "most_frequent" strategy) may have diluted the specific feature combinations required for accurate classification in the minority classes.

5.3 Implications for Deployment

For a real-world application at a scale of 100,000 records, the current training set is insufficient. To reach production-grade metrics, the pipeline would require a significantly larger and more diverse training corpus that includes the edge cases introduced by the missing value simulation. Because even best algorithm like **XGBoost** underperformed. Figure 3

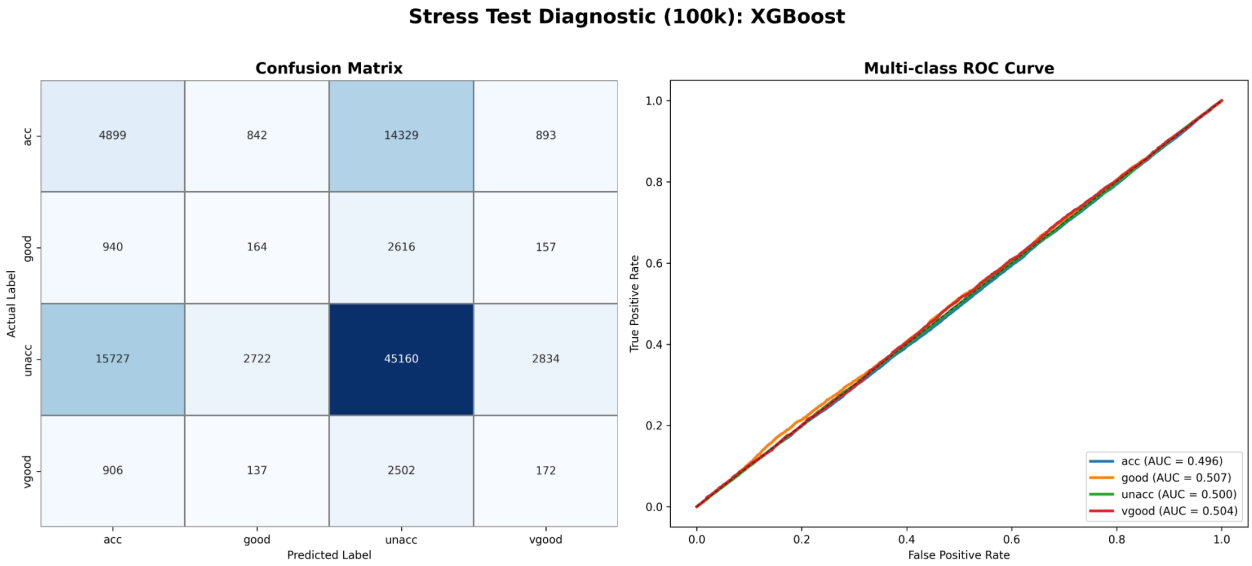


Figure 3: Diagnostic Card for XGBoost on 100k Records: Confusion Matrix (Left) and Multi-class ROC Curve (Right).

6 Appendix

6.1 Source Code - GitHub

GitHub

6.2 YouTube Video

YouTube Video

6.3 Chat Session Links

Gemini

DeepSeek

6.4 Tables

Table 4: Experimental Hardware Configuration

Category	Configuration
CPU	Intel Core i9-13900K processor with 24 cores (8P+16E), boost frequency up to 5.8 GHz, 36 MB Intel Smart Cache
RAM	32 GB DDR5 memory operating at 4800 MT/s
GPU	NVIDIA GeForce RTX 4070 with 8 GB GDDR6X VRAM, 5888 CUDA cores, 192-bit memory interface
OS	Zorin OS 16.3 (Ubuntu 20.04 LTS based) Linux distribution [1]

6.5 Additional Plots

Table 5: Performance Metrics for KNN Models (Mean Values Only)

Model	F1 Score	Accuracy	Precision	Recall
KNN_3	0.842083	0.842083	0.842083	0.842083
KNN_15	0.840557	0.840557	0.840557	0.840557
KNN_13	0.839813	0.839813	0.839813	0.839813
KNN_14	0.836940	0.836940	0.836940	0.836940
KNN_17	0.835477	0.835477	0.835477	0.835477
KNN_1	0.833665	0.833665	0.833665	0.833665
KNN_16	0.831958	0.831958	0.831958	0.831958
KNN_11	0.828496	0.828496	0.828496	0.828496
KNN_12	0.825362	0.825362	0.825362	0.825362
KNN_18	0.824218	0.824218	0.824218	0.824218
KNN_19	0.822100	0.822100	0.822100	0.822100
KNN_20	0.818519	0.818519	0.818519	0.818519
KNN_7	0.818234	0.818234	0.818234	0.818234
KNN_9	0.816794	0.816794	0.816794	0.816794
KNN_8	0.815496	0.815496	0.815496	0.815496
KNN_10	0.812411	0.812411	0.812411	0.812411
KNN_5	0.809472	0.809472	0.809472	0.809472
KNN_4	0.790333	0.790333	0.790333	0.790333
KNN_6	0.789160	0.789160	0.789160	0.789160
KNN_2	0.773613	0.773613	0.773613	0.773613

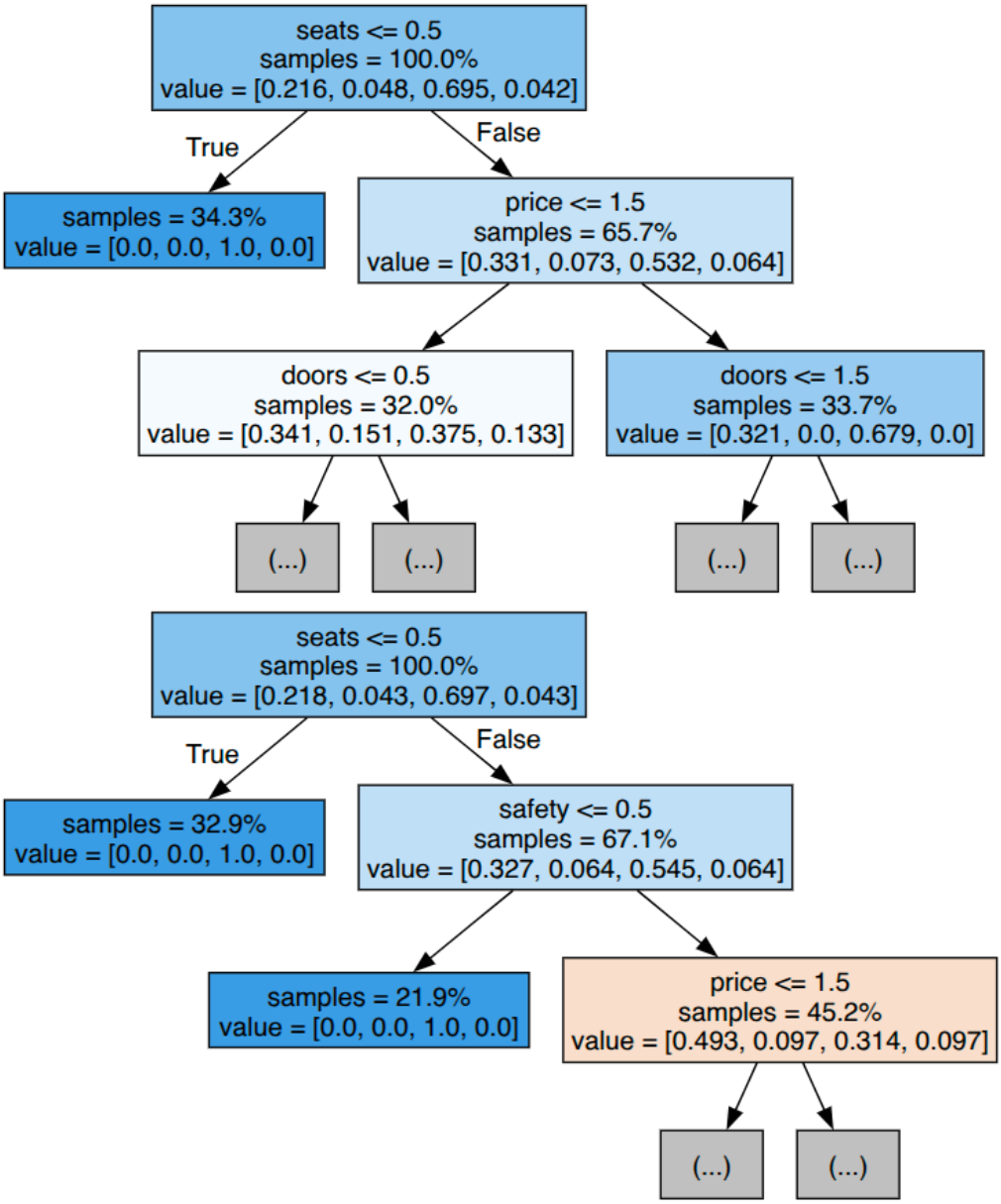


Figure 4: Decision Free by splitting features

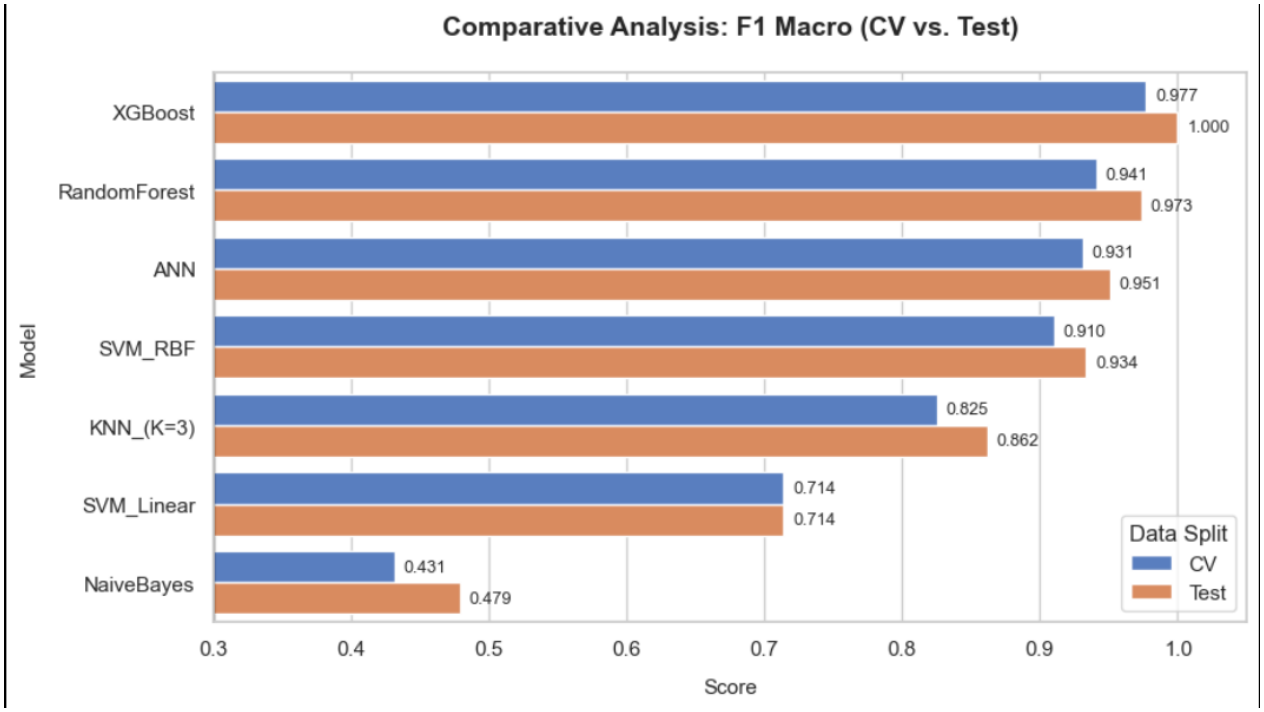


Figure 5: F1 Comparisons

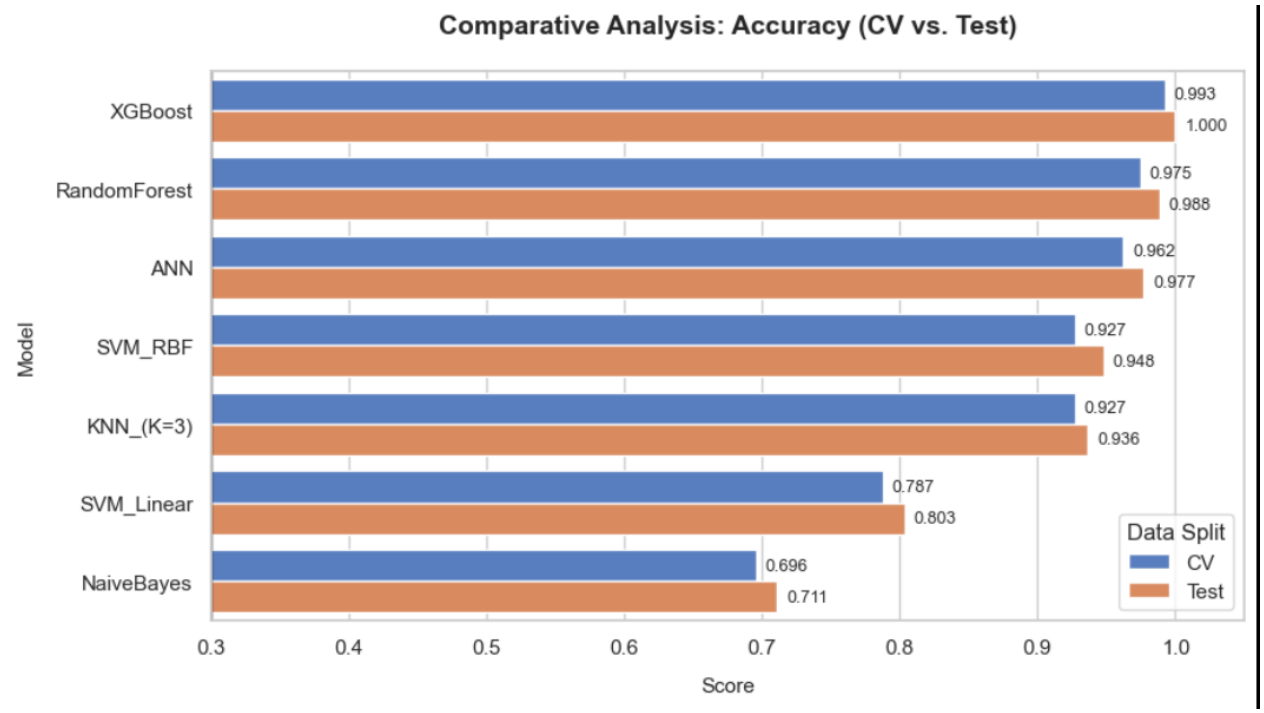


Figure 6: Accuracy Comparisons

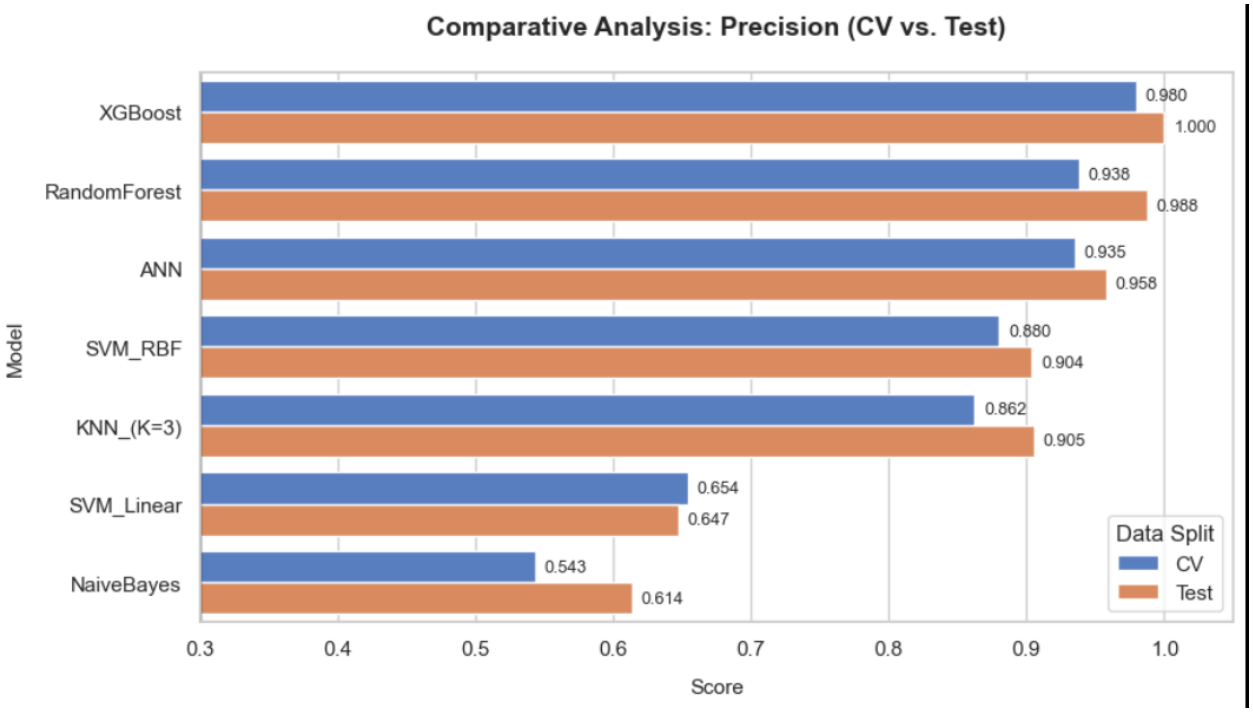


Figure 7: Precision Comparisons

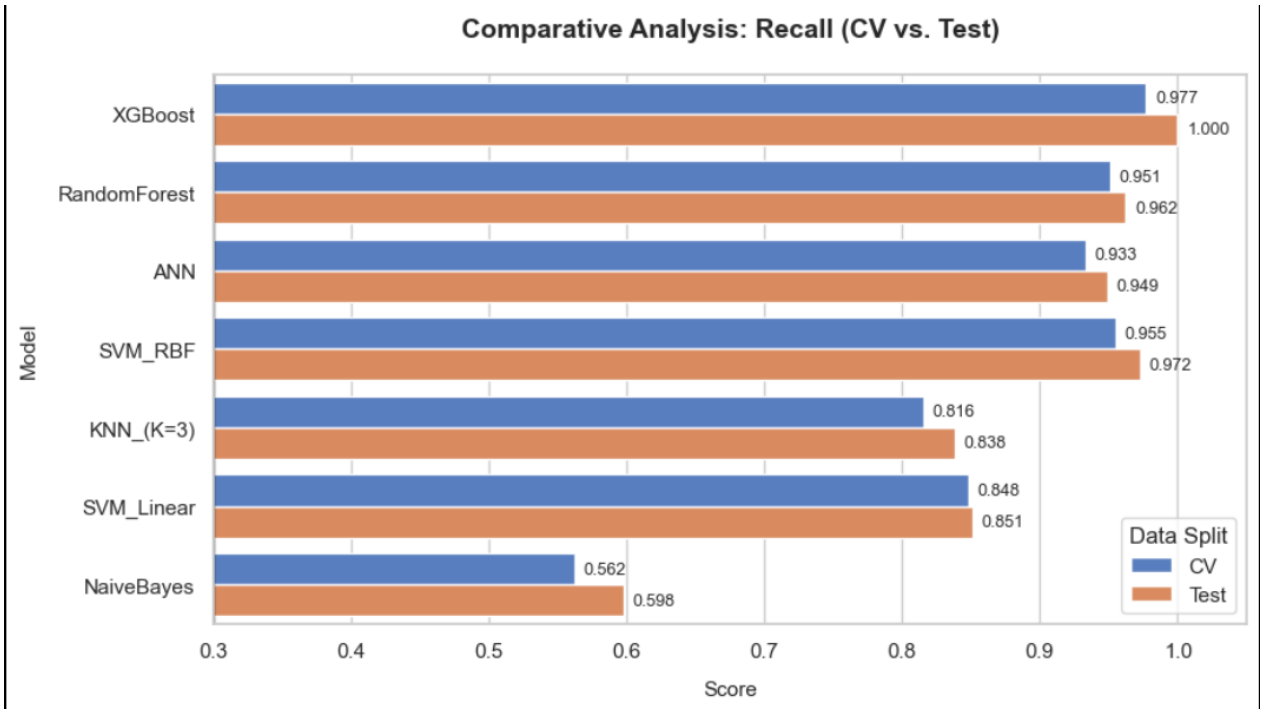


Figure 8: Recall Comparisons

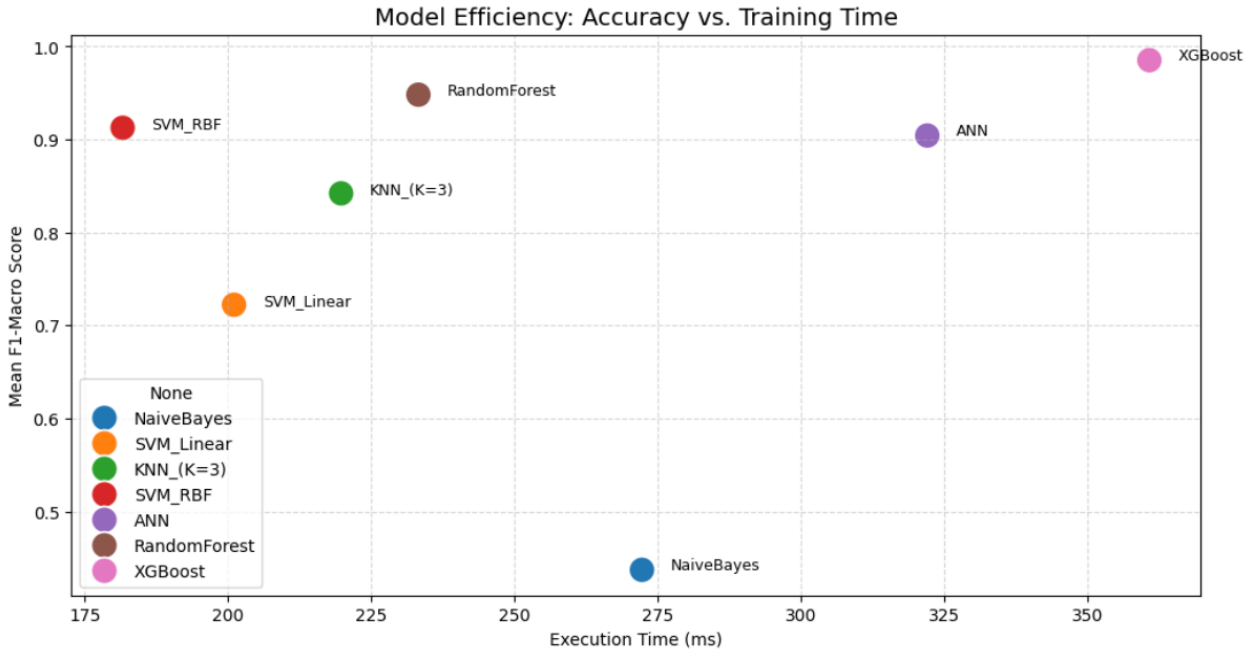


Figure 9: Performance: F1 vs Time

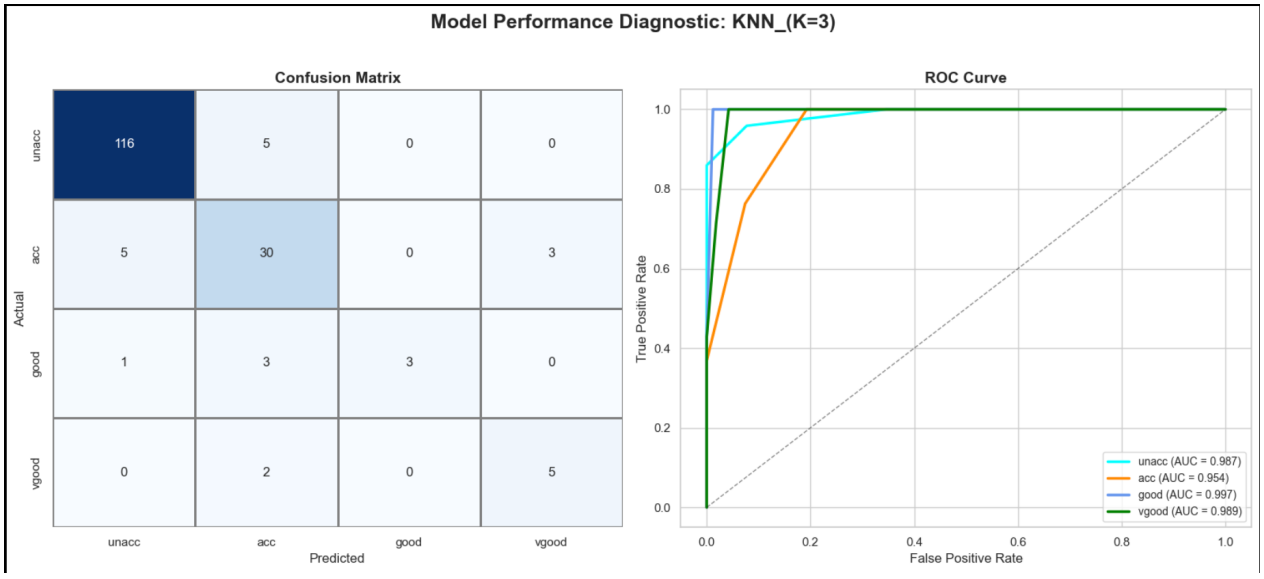


Figure 10: KNN k = 3

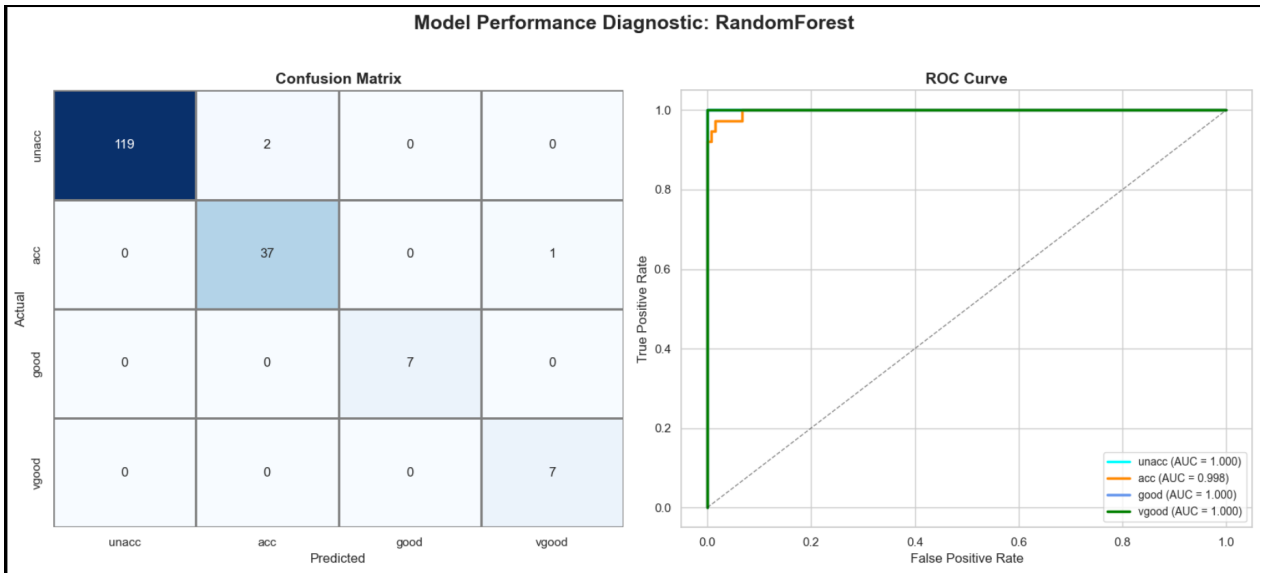


Figure 11: Random Forest

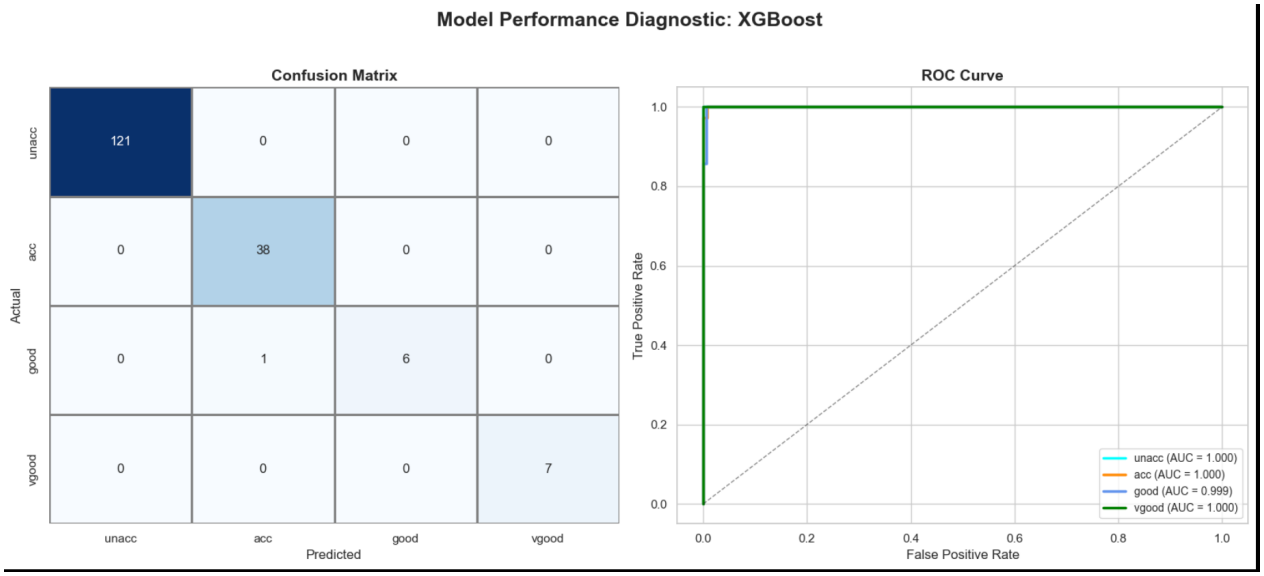


Figure 12: XG Boost

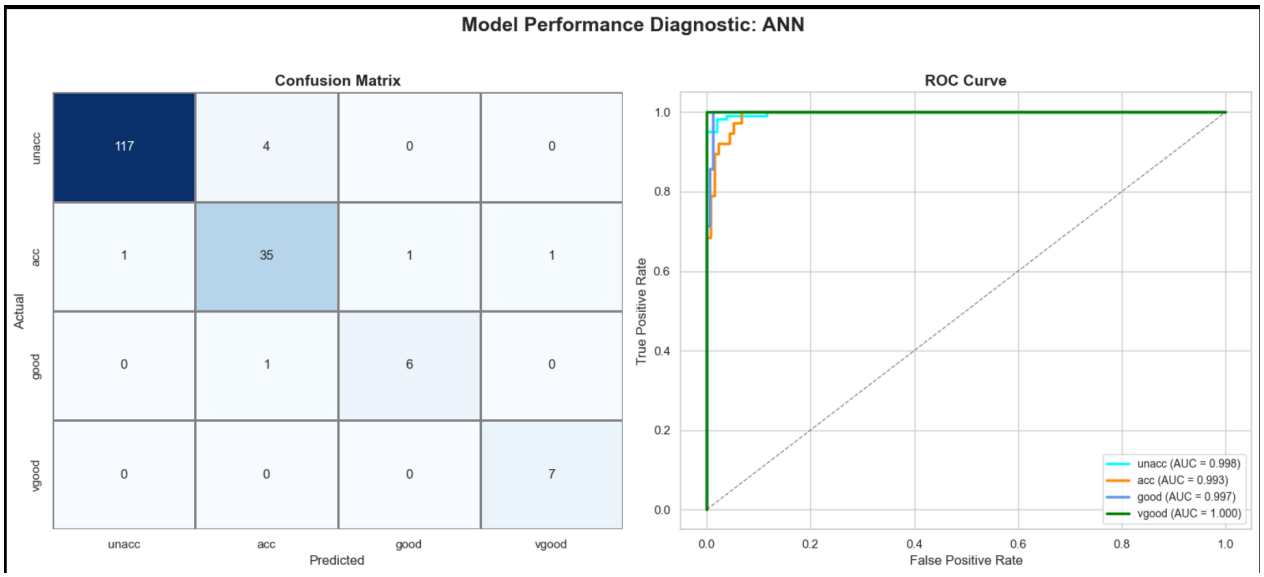


Figure 13: ANN

Bibliography