AAE 718 Worksheet 11

Caroline Kuo

June 12, 2025

Problem 1: Drug Classification using Decision Tree

This task involves predicting which drug a patient is likely to be prescribed based on their demographic and basic medical information. I developed two decision tree models: a basic model using four core features and an enhanced version using all features and applying SMOTE to address class imbalance.

The dataset `drug200.csv` contains 200 records and five features: Age, Sex, Blood Pressure (BP), Cholesterol, and Sodium-to-Potassium ratio (Na_to_K). The target variable is Drug, which includes five possible categories: drugA, drugB, drugC, drugX, and drugY.

For Model A, I used only Age, Sex, BP, and Cholesterol. I did not apply any resampling, and I used an 80/20 train-test split. This model achieved 93.75% training accuracy and 60% testing accuracy. It showed limited generalization capacity, as reflected in the classification report and confusion matrix.

In contrast, Model B included all features and applied SMOTE to balance the training data. I also limited the tree depth to 5 to reduce overfitting. This model significantly improved the performance, reaching a testing accuracy of 95%. Its classification report showed high precision and recall across all drug categories, but the unusually high scores raise concerns about possible overfitting.

Overall, while Model A adheres to the original feature constraints and reflects a realistic approach, Model B demonstrates how additional preprocessing and feature expansion can improve accuracy. However, results should be further validated through cross-validation or with a larger dataset to ensure robustness.

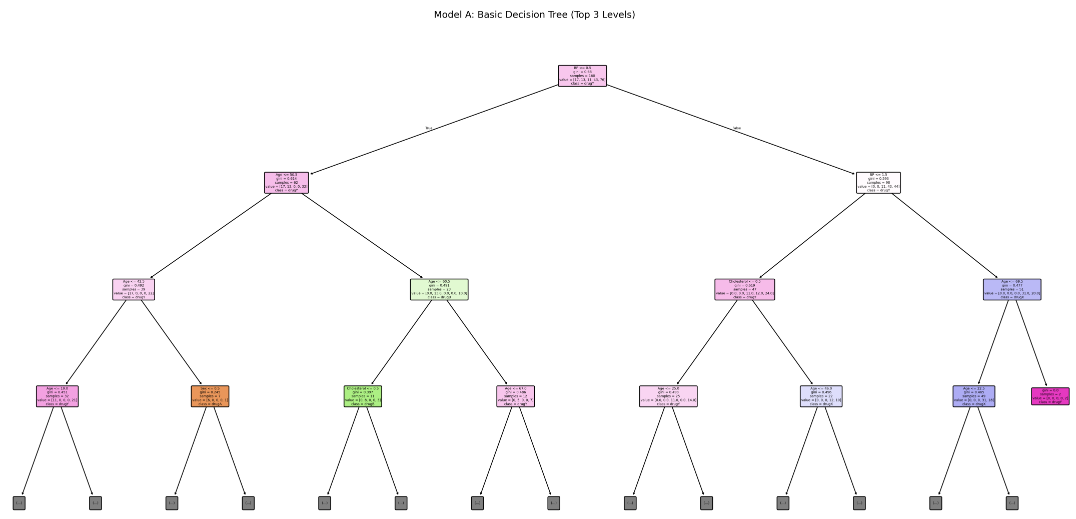


Figure 1: Model A: Basic Decision Tree

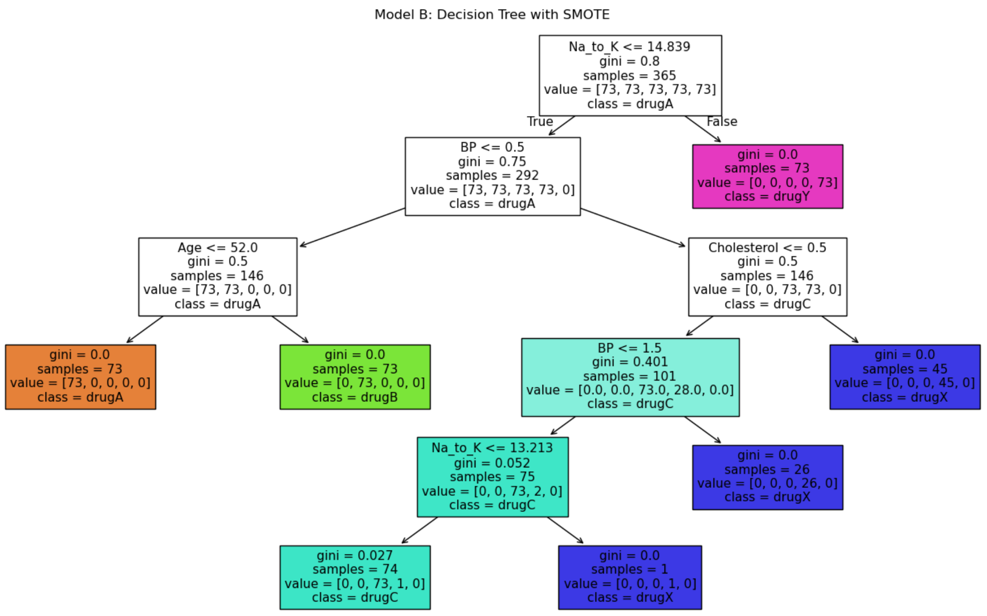


Figure 2: Model B: Decision Tree with SMOTE (Top Levels)

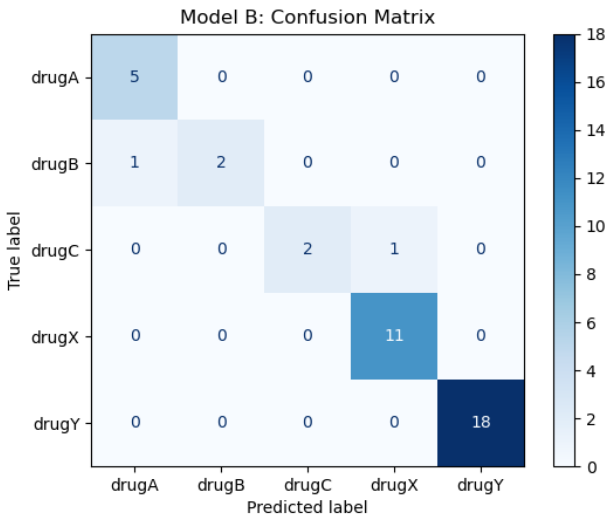


Figure 3: Model B: Confusion Matrix

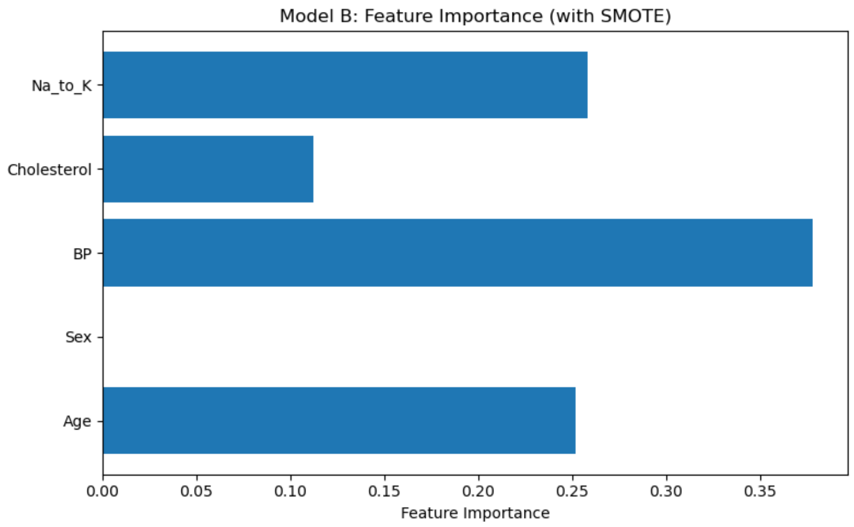


Figure 4: Model B: Feature Importance

Problem 2: Default Prediction with Multiple Models

This task focuses on predicting credit card default using the UCI ”Default of credit card clients” dataset. I implemented and compared three classification models: logistic regression, decision tree, and neural network, each with a baseline and a tuned version.

Logistic Regression (LR)

The baseline logistic regression model used standardized numerical features and default hyperparameters. With an 80/20 train-test split stratified by the target variable, the model achieved an accuracy of 81% and a log loss of 0.4635. However, its recall for defaulters was only 24%, indicating difficulty in detecting the minority class.

Using GridSearchCV, I tuned parameters including `C`, `penalty`, and `solver`. The tuned model had very similar performance: 80.98% accuracy and the same log loss, with minimal improvement in recall. This suggests that the default settings were already near-optimal and that the imbalance issue remains unresolved.

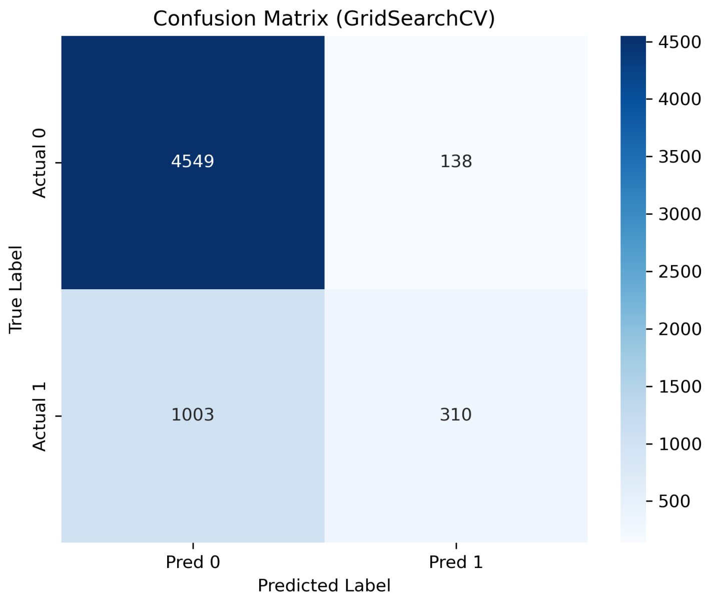


Figure 5: Logistic Regression (Tuned): Confusion Matrix

Decision Tree (DT)

The basic decision tree used default settings and achieved 72.15% accuracy with a very high log loss (over 10), which pointed to instability. After tuning `max_depth`, `min_samples_split`, and `min_samples_leaf` using GridSearchCV, the tuned tree reached 81.92% accuracy and a much lower log loss of 0.4675. It improved precision and recall for the default class, especially precision (from 0.37 to 0.65), although recall remained moderate.

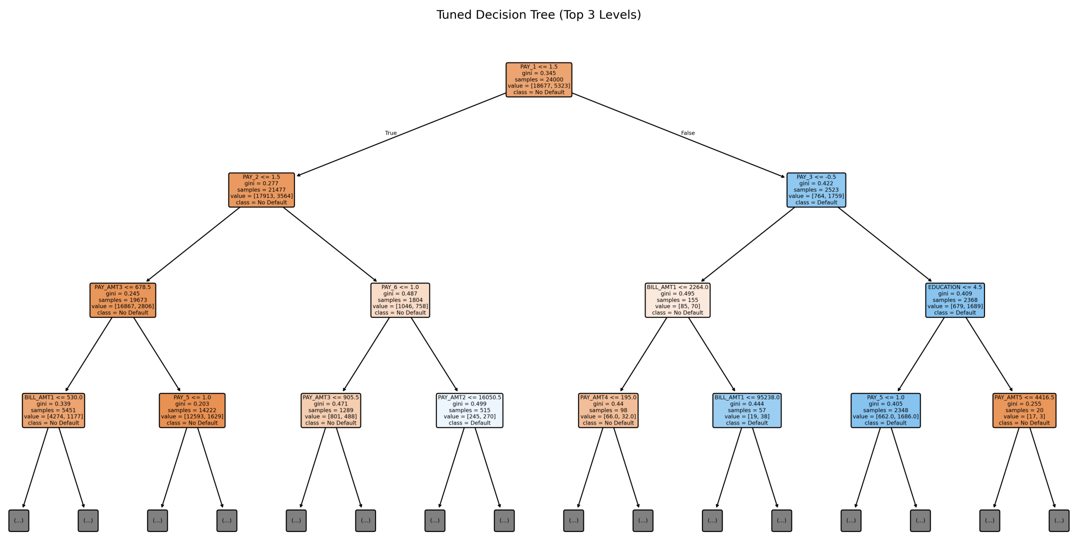


Figure 6: Tuned Decision Tree (Top 3 Levels)

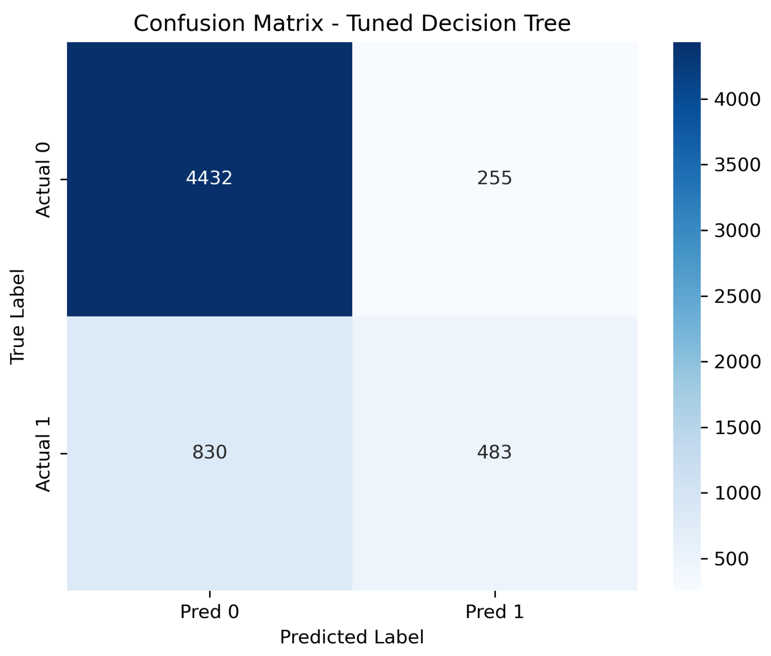


Figure 7: Confusion Matrix: Tuned Decision Tree

Neural Network (NN)

I implemented a baseline multilayer perceptron with one hidden layer of 100 units. This model achieved 81.23% accuracy and a log loss of 0.4563. After tuning `hidden_layer_sizes`, `alpha`, `learning_rate_init`, and `solver`, the performance slightly improved to 81.67%

accuracy and a log loss of 0.4446. Recall for defaulters improved marginally but remained under 40%, showing that additional strategies like class weighting or SMOTE might still be needed.

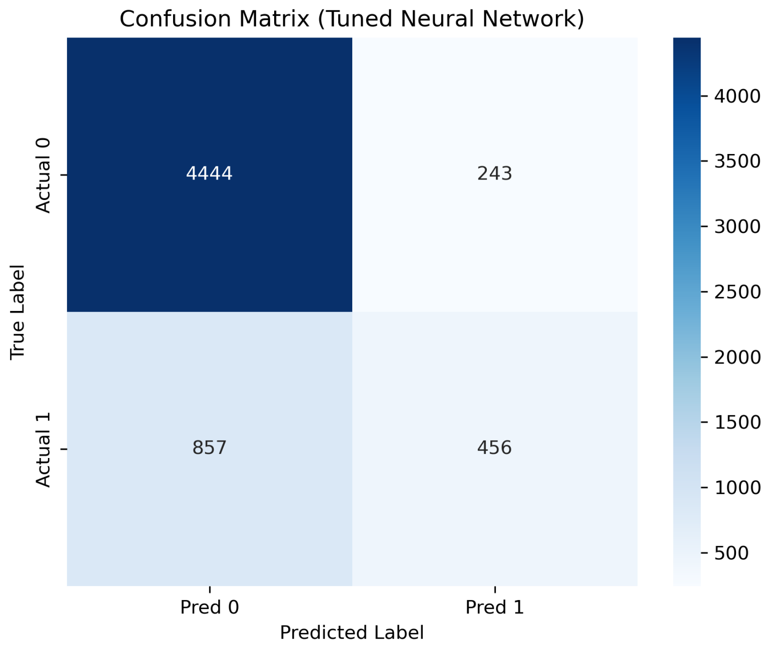


Figure 8: Confusion Matrix: Tuned Neural Network

Problem 3: Titanic Survival Prediction with XGBoost and Optuna

In this task, I developed a survival prediction model for Titanic passengers using XGBoost, with Optuna for hyperparameter tuning. The features included Pclass, Sex, Age, SibSp, Parch, Fare, and Embarked. I applied label encoding for categorical variables and mean imputation for missing values in Age and Fare.

After preprocessing, I split the data into training and validation sets. Optuna was used to perform 50 trials of parameter search, evaluating combinations of tree depth, learning rate, sampling ratios, and regularization. The best parameters were then used to train a final XGBoost model.

On the validation set, the model achieved 80% accuracy. The confusion matrix and classification report showed a precision of 0.80 for both classes, recall of 0.88 for non-survivors and 0.69 for survivors, with F1-scores of 0.84 and 0.74 respectively. The model was then applied to the test dataset, and the output was saved as `titanic_prediction_optuna_xgb_smote.csv` for submission or further evaluation.

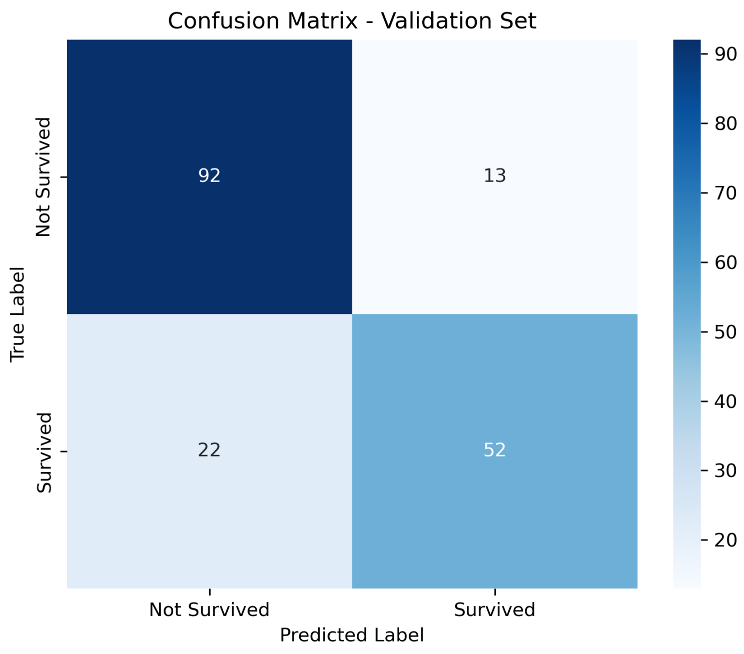
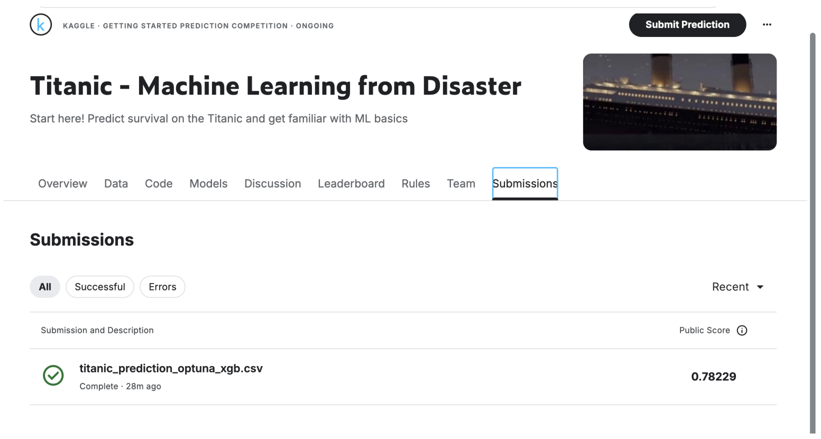


Figure 9: Confusion Matrix: Titanic Validation Set



Titanic - Machine Learning from Disaster

Start here! Predict survival on the Titanic and get familiar with ML basics

Submissions

Submission and Description	Public Score
titanic_prediction_optuna_xgb.csv Complete · 28m ago	0.78229

Figure 10: Competition Submission