Applied Analytics Project

Analyzing US Accident Data to Predict High-Risk Areas and Times in Massachusetts

Week 8 – XGBoost Model Variations for Multi-Class Classification

Major: Applied Analytics

Name: Gefan Wang, Chenhe Shi, Tianchen Liu

Date: 03.21.2025

1. Chosen Model and Rationale

Among several candidate models (K-Nearest Neighbors, Decision Tree, Random Forest, XGBoost, Naive Bayes, and PCA + Logistic Regression), we selected **XGBoost** as our this week approach. This choice was driven by several factors:

1. **Data Characteristics**: Our dataset contained numerous correlated features, including numerical variables (e.g., temperature, precipitation, visibility) and categorical variables related to weather conditions and time. XGBoost's ability to handle mixed data types and capture complex feature interactions made it a strong choice.
2. **Imbalance in Target Classes**: Our dataset exhibited class imbalance, with some categories (e.g., major crashes with serious injuries or fatalities) representing a small fraction of total observations. To address this, we used **`scale_pos_weight`** to rebalance the classes and improve model learning.
3. **Model Performance and Optimization**: XGBoost provides built-in regularization (**`reg_alpha`** and **`reg_lambda`**) to prevent overfitting, along with powerful hyperparameter tuning options that help optimize performance.
4. **Interpretability and Efficiency**: XGBoost offers feature importance rankings, making it easier to analyze which factors contribute most to predictions. Additionally, it is computationally efficient compared to some deep learning approaches.

2. Model Complexity

The complexity of our modeling approach is primarily influenced by:

1. The number of estimators (**`n_estimators`**) which determines the boosting iterations.
2. Tree depth (**`max_depth`**) which affects how intricate the decision boundaries become.
3. Learning rate (**`learning_rate`**), controlling the step size of updates.
4. Regularization terms (**`reg_alpha`** and **`reg_lambda`**) to prevent overfitting.
5. Class balancing with **`scale_pos_weight`** to address the dataset's imbalanced distribution.

3. Hyperparameter Evaluation

To optimize model performance, we tested three variations:

Variation	max_depth	learning_rate	n_estimators	scale_pos_weight	reg_alpha	reg_lamda
1	3	0.05	150	balanced	0.1	0.1
2	4	0.1	200	balanced	0.2	0.2
3	5	0.05	200	0.156	0.1	0.3

These variations were chosen to assess the trade-off between model complexity and generalization by adjusting tree depth, learning rate, and regularization strength.

4. Model Performance Metrics

We used the following metrics to evaluate model performance:

1. **Accuracy**: Measures overall correctness of predictions.
2. **Log Loss**: Captures the probability-based error for misclassified samples.
3. **AUC (Area Under the Curve)**: Evaluates the model's ability to distinguish between classes, particularly useful for imbalanced datasets.
4. **Precision, Recall, and F1-Score**: Provide insight into per-class classification performance.

5. Metrics Calculation and Comparison

We trained and evaluated our model using training and validation datasets. The table below summarizes the results across the three variations:

Variation	Training Accuracy	Validation Accuracy	Training Log Loss	Validation Log Loss	Training AUC	Validation AUC
1	0.7262	0.7235	0.5853	0.5982	0.8562	0.8305
2	0.7874	0.7578	0.4635	0.5220	0.9303	0.8863
3	0.7324	0.7283	0.5647	0.5826	0.8739	0.8469

6. Analysis of Training vs. Validation Metrics

1. Training accuracy across variations remained relatively stable, suggesting well-optimized training behavior.
2. Validation accuracy ranged between **0.72-0.76**, indicating good generalization across models.
3. Log loss values were slightly higher in validation compared to training, reflecting mild overfitting.
4. AUC values showed strong classification performance, with **Variation 2 achieving the highest Validation AUC (0.8863).**

7. Best Model Selection

Based on our results, **Variation 2 (`max_depth=4, learning_rate=0.1, n_estimators=200`)** was selected as the best model for the week. This model achieved:

1. The highest Validation AUC (**0.8863**), indicating superior class separation.
2. Competitive validation accuracy (**0.7578**), ensuring strong predictive performance.
3. Lower validation log loss (**0.5220**), suggesting better probabilistic predictions.
4. More balanced precision-recall scores across classes.

8. Conclusion

Our experiment demonstrated that XGBoost with regularization and class balancing significantly improves multi-class classification performance. The best model achieved robust validation results while avoiding overfitting. Future work could explore additional feature engineering and automated hyperparameter tuning to further optimize performance.

9. Reference

[1] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785-794.