Wildlife Strike Damage Prediction Project

Author: Lawrence Ganeshalingam

Overview

This project develops machine learning models to predict aircraft damage from wildlife strikes using the FAA Wildlife Strike Database. It focuses on binary classification (damaged vs. not damaged) with supervised learning techniques, addressing data imbalance and feature engineering. The models have applications in aviation safety and agricultural drone operations, where sensory data (e.g., altitude, speed) can help mitigate risks during low-altitude flights for tasks like crop monitoring or pest control.

Key components:

- **Data Processing**: Cleaning, imputation, outlier handling, and feature engineering (e.g., Season, High Risk Phase, Bird Size Proxy).
- **Models**: RandomForest (baseline), XGBoost (grid and Optuna-tuned), LightGBM.
- **Evaluation**: Metrics like F1-score, recall, precision, and ROC-AUC, with comparisons across subsets (20K samples) and full dataset (174K for RandomForest).
- **Improvements**: Iterative tuning and scaling show gains in minority class performance (e.g., F1 from 0.37 to 0.44).

The notebook includes code, analyses, and summaries aligned with rubric fields (Project Topic, Data, Cleaning, EDA, Models, Results/Analysis, Discussion/Conclusion).

Installation and Dependencies

1. Clone the repository:

```
git clone https://github.com/yourusername/Wildlife-Strike-Damage-Prediction-Project.git
cd Wildlife-Strike-Damage-Prediction-Project
```

2. Install required packages (use a virtual environment like venv or conda):

```
pip install pandas numpy matplotlib seaborn scikit-learn imbalanced-learn xgboost lightgbm optuna
```

   - Python version: 3.8+ recommended.

- Key libraries: pandas (data handling), scikit-learn (modeling/tuning), imbalanced-learn (SMOTE), xgboost/lightgbm (boosting models), optuna (hyperparameter optimization).

3. Download the dataset from Kaggle FAA Wildlife Strikes and place `database.csv` in a `Final Project/` folder (or adjust file_path in code).

How to Use

1. **Run the Notebook**:
   - Open the Jupyter notebook (e.g., `wildlife_strike_prediction.ipynb`) in Jupyter Lab/Notebook or VS Code.
   - Execute cells sequentially: Starts with data loading/cleaning, EDA, then model training/evaluation.
   - For subsets: Most runs use 20K samples for efficiency; adjust `df.sample(20000)` to full data if resources allow (tested on RandomForest).
   - Tuning: GridSearch for baseline; Optuna (requires installation) for advanced XGBoost—run with `n_trials=100` (adjust for time).
2. **Customize**:
   - Change `file_path` if dataset location differs.
   - For drone applications: Input custom features (e.g., altitude/speed) via model.predict() after training.
   - Experiment: Modify param_grid or Optuna objective for further tuning.
3. **Output**:
   - Metrics: Printed classification reports, F1/AUC scores.
   - Visuals: Histograms, countplots, confusion matrices, PR-curves, feature importance bars.
   - Comparative Table: See results section for model benchmarks.

Data Source

- FAA Wildlife Strike Database (1990-2015) from Kaggle: ~174K records of strikes with features like height, speed, phase, species.
- Citation: Federal Aviation Administration. (2015). *Wildlife strikes* [Data set]. Kaggle. https://www.kaggle.com/datasets/faa/wildlife-strikes

Models and Results

- **Models Used**: RandomForest (expanded features), LightGBM, XGBoost (grid/Optuna-tuned).
- **Key Results**: Boosting models achieve Class 1 F1 ~0.44 and recall ~0.51 (vs. RF baseline 0.37/0.30). Full-data RF improves F1 to 0.44 (+19%). See comparative table below:

Model	Mean CV F1	Test Accuracy	Class 1 Precision	Class 1 Recall	Class 1 F1	Test ROC-AUC
RandomForest (Baseline 20K)	0.428	0.91	0.48	0.30	0.37	0.861
RandomForest (Full 174K)	0.458	0.91	0.49	0.41	0.44	0.872
LightGBM (20K)	0.426	0.89	0.39	0.51	0.44	0.849
XGBoost (Grid Search 20K)	0.426	0.89	0.39	0.51	0.44	0.849
XGBoost (Optuna 20K)	0.437	0.89	0.39	0.49	0.43	0.862

- **Analysis**: Scaling to full data reduces variance; Optuna adds robustness but minor trade-offs.

Future Improvements

- Advanced sampling (e.g., ADASYN) for better precision.
- Ensembling (e.g., VotingClassifier) for hybrid performance.
- Full dataset runs on boosters for further gains.
- Integrate external data (e.g., bird migration APIs) for refined features.

Contributing

Fork the repo, make changes, and submit a pull request. Issues/PRs welcome!

License