Bird Strikes in Aviation Prediction Bayesian Logical Regression

Zhanxu Liu
ISYE 6420

November 23, 2024

Introduction

Bird strikes pose a significant risk to aviation safety, with potential consequences ranging from minor inconveniences to severe aircraft damage and financial losses. The phenomenon of bird strikes occurs when an aircraft collides with one or more birds during flight, often resulting in operational disruptions. Over the years, the frequency of bird strikes has risen due to increasing air traffic and expanding wildlife populations near airports.

Understanding the factors contributing to bird strike incidents and their outcomes is crucial for mitigating risks and improving aviation safety. While not all bird strikes result in damage, identifying the conditions under which damage is more likely can help inform preventative measures. This project leverages a publicly available dataset from Kaggle on bird strikes to predict the likelihood of aircraft damage, focusing on the binary outcome of whether the aircraft sustained damage ("Caused damage" or "No damage").

Using Bayesian logistic regression, this analysis aims to quantify the relationship between various predictors—such as the type of aircraft, altitude, location, and time of day—and the probability of aircraft damage. Unlike traditional frequentist approaches, the Bayesian framework provides a probabilistic interpretation of model parameters, allowing for the incorporation of prior knowledge and the generation of credible intervals for predictions.

The results of this study will provide actionable insights into the factors influencing damage severity in bird strike incidents. These findings can help aviation authorities, airlines, and airport management enhance bird strike mitigation strategies, ultimately contributing to safer skies.

Logistic Regression

This is a logistic regression model. We consider each incident as a single Bernoulli event.

$$y_i \mid \boldsymbol{\beta}, \boldsymbol{x}_i \sim \text{Bernoulli}(p_i)$$

$$g(p_i) = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}$$

$$\beta_0 \sim \mathcal{N}(0, \sigma_0^2)$$

$$\beta_j \sim \mathcal{N}(0, \sigma^2), \quad j = 1, 2, \ldots, k$$

Bayesian Model Overview

The goal is to model the probability p of a binary outcome y (e.g., damage/no damage) using predictors $\boldsymbol{X}$ and Bayesian priors on the model parameters.

General Logistic Regression Formula

$$p(y = 1 \mid \boldsymbol{X}, \boldsymbol{\beta}, \text{intercept}) = \frac{1}{1 + \exp\left(-(\text{intercept} + \boldsymbol{X}\boldsymbol{\beta})\right)}$$

Priors

- Intercept Prior (α):

  $$\alpha \sim \mathcal{N}(0, \tau_{\text{intercept}})$$

  where $\tau_{\text{intercept}} = 0.1$.

- Coefficients Prior (β):

  $$\beta_j \sim \mathcal{N}(0, \tau_{\text{betas}})$$

  for each coefficient j, where $\tau_{\text{betas}} = 0.05$ and the dimension of β matches the number of predictors in $\boldsymbol{X}$.

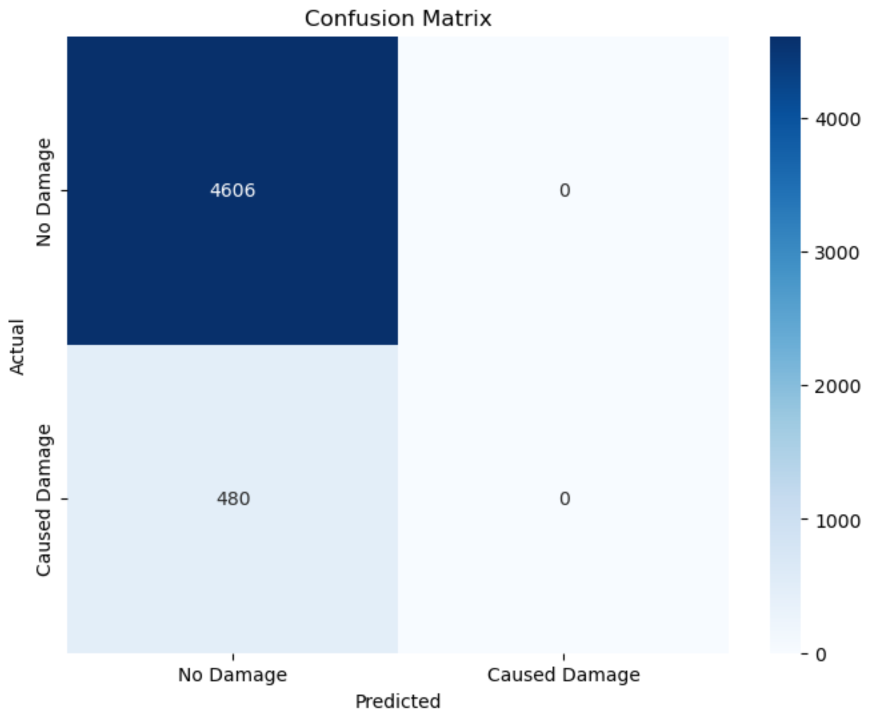
Naive Baseline

In order to evaluate the regression model, 20% of the dataset is randomly selected as the testing set. The remaining 80% is used as the training set.

To determine whether machine learning is suitable for our problem, we first establish a baseline against which to compare our results. For classification tasks, a simple naive baseline involves predicting the most frequent class in the training data for all testing observations. If our model fails to outperform this baseline, it may indicate that machine learning is not appropriate for the task, or that an alternative modeling approach should be considered.

```
Accuracy of Model: 90.56%
F1 Score of Model: 0.0000
```

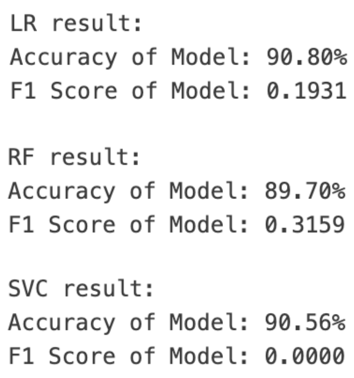
If our model fails to exceed an accuracy of 90.56%, we will need to reconsider our approach. However, it is important to note that this baseline is limited in its usefulness, as it does not identify any cases of diabetes (outcome of 1). This limitation is evident in the confusion matrix.



While accuracy appears acceptable (because of the high number of true negatives), the model is unable to detect any cases of Caused Damage, making it ineffective for this use case.

Evaluate Standard Machine Learning Methods

To compare performance, we will implement three standard machine learning models (logistic regression,). Hyperparameter tuning will not be performed in this analysis, which means there is potential for improved performance through optimization. However, the primary focus here is to evaluate the baseline performance of these models relative to Bayesian Logistic Regression.



```
LR result:
Accuracy of Model: 90.80%
F1 Score of Model: 0.1931

RF result:
Accuracy of Model: 89.70%
F1 Score of Model: 0.3159

SVC result:
Accuracy of Model: 90.56%
F1 Score of Model: 0.0000
```

These results indicate that only the logistic regression model is acceptable for this task. Additionally, based on the F1 score, logistic regression and random forest outperform the baseline. We can compare these results with the Bayesian model that will be constructed next.

Bayesian Logistic Regression

The first step is to define our model. In the frequentist interpretation of Logistic Regression, the natural logarithm of the odds of the outcome, known as the logits, is modeled as a linear combination of the input features. Additionally, an error term is included to account for random sampling noise or unobserved factors. The formula for the frequentist interpretation is as follows:

$$\begin{aligned}\text{logit} = \beta_0 &+ \beta_1(\text{NumberStruckActual}) + \beta_2(\text{Altitude}) + \beta_3(\text{WildlifeSize}) \\ &+ \beta_4(\text{FlightPhase}) + \beta_5(\text{ConditionsSky}) + \beta_6(\text{PilotWarned} + \beta_7(\text{IsAircraftLarge})\end{aligned}$$

The log odds can then be converted to a probability of the output:

$$p = \frac{1}{1 + e^{-\text{logit}}}$$

For our problem, we are interested in finding the probability that a aircraft was damaged:

$$p(\text{Caused damage} \mid \text{Features})$$

The goal is to estimate the "optimal" β values, which typically represent the most likely parameters given the observed inputs and outputs. Logistic Regression determines these β values through an iterative process known as Maximum Likelihood Estimation.

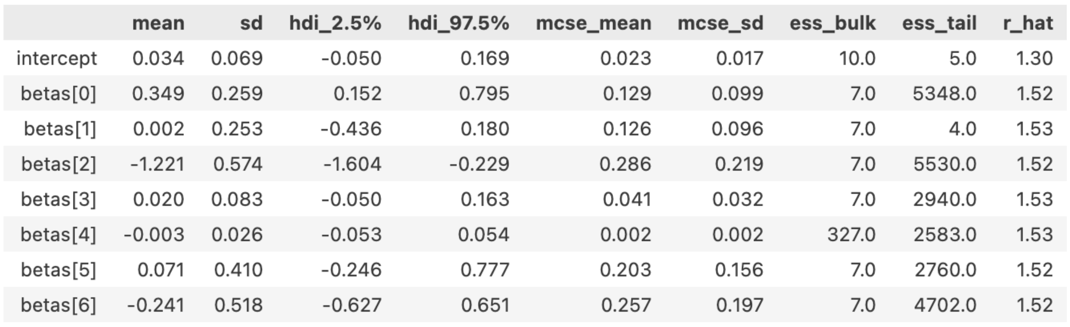
On the other hand, the Bayesian framework models the likelihood of the data as being generated from a probability distribution. For binary classification, this likelihood is described by a Bernoulli distribution:

$$p(\text{Caused damage} \mid \text{Features}) = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i}$$

The objective is to find the posterior probability distribution of the model parameters given the inputs and outputs:

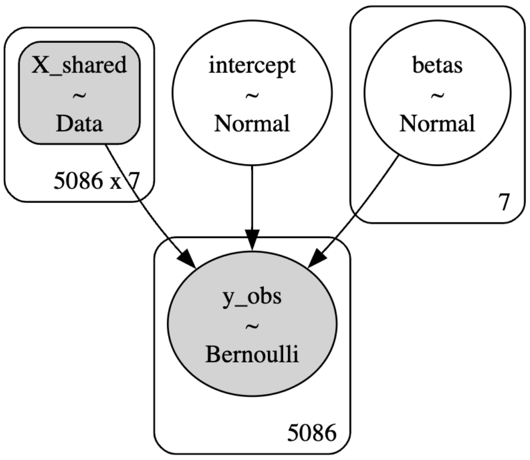
$$P(\boldsymbol{\beta} \mid X, y) = \frac{P(y \mid \boldsymbol{\beta}, X)P(\boldsymbol{\beta} \mid X)}{P(X, y)}$$

Priors: if we have information about the model parameters, we can inject it into the model. Posterior distributions: the result is a distribution for the model parameters instead of a point estimate. **Summary Statistics**



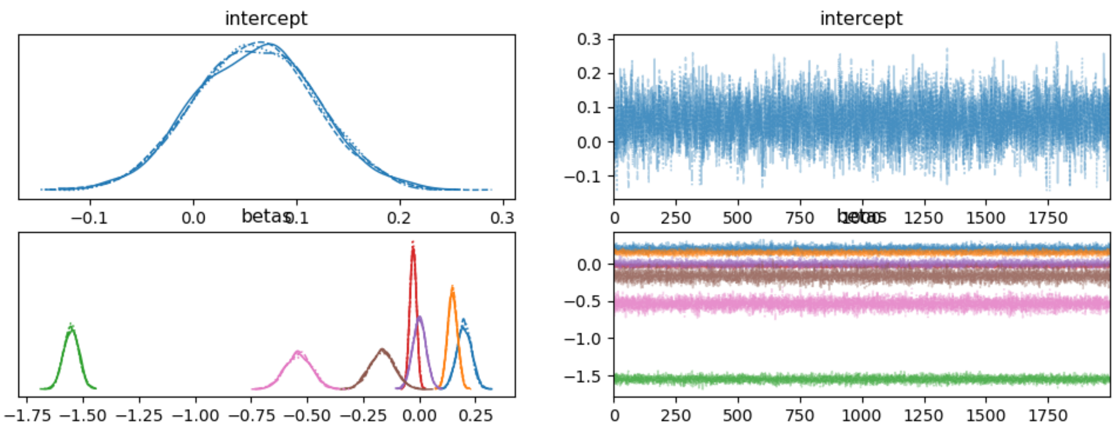
	mean	sd	hdi_2.5%	hdi_97.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
intercept	0.034	0.069	-0.050	0.169	0.023	0.017	10.0	5.0	1.30
betas[0]	0.349	0.259	0.152	0.795	0.129	0.099	7.0	5348.0	1.52
betas[1]	0.002	0.253	-0.436	0.180	0.126	0.096	7.0	4.0	1.53
betas[2]	-1.221	0.574	-1.604	-0.229	0.286	0.219	7.0	5530.0	1.52
betas[3]	0.020	0.083	-0.050	0.163	0.041	0.032	7.0	2940.0	1.53
betas[4]	-0.003	0.026	-0.053	0.054	0.002	0.002	327.0	2583.0	1.53
betas[5]	0.071	0.410	-0.246	0.777	0.203	0.156	7.0	2760.0	1.52
betas[6]	-0.241	0.518	-0.627	0.651	0.257	0.197	7.0	4702.0	1.52

The posterior means and 95% highest density intervals (HDI) provide insight into the central tendencies and credible ranges of the parameters. For example, Beta[2] shows a strong negative mean (-1.221) with an HDI entirely below zero, indicating a likely negative association with the outcome. Other parameters, such as Beta[4], have narrow HDIs near zero, suggesting minimal influence. While effective sample sizes (ess_bulk and ess_tail) are generally sufficient, the r_hat values exceed 1.5 for all parameters, indicating potential convergence issues that need to be addressed. These findings highlight key parameter effects while also suggesting the need for further model refinement and improved sampling convergence.



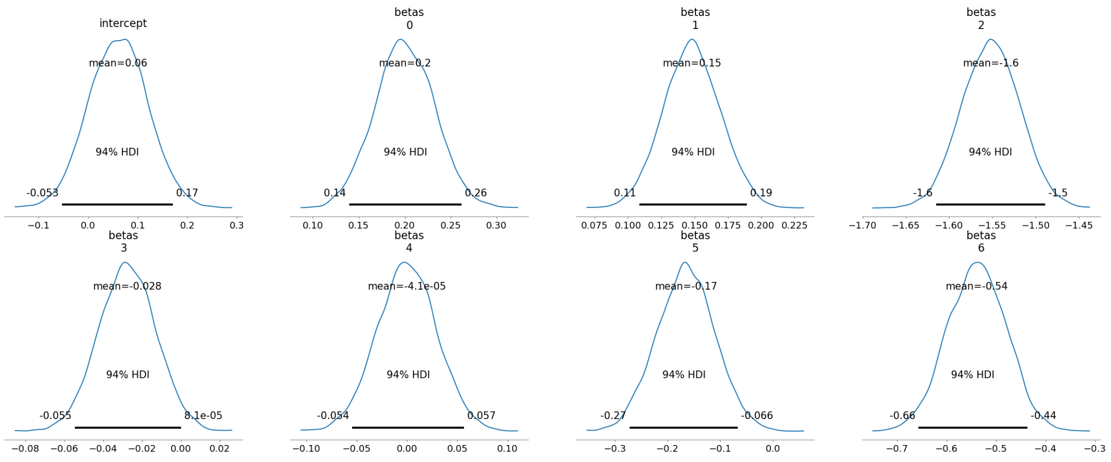
Density and trace plots

Python's PyMC package was used with 2,000 samples each and 1,000 tuned.



Posterior Plot of the Variables

The posterior plot of the variables represents the posterior distributions for each parameter in the model. The distributions are expected as normal from the priors I set.



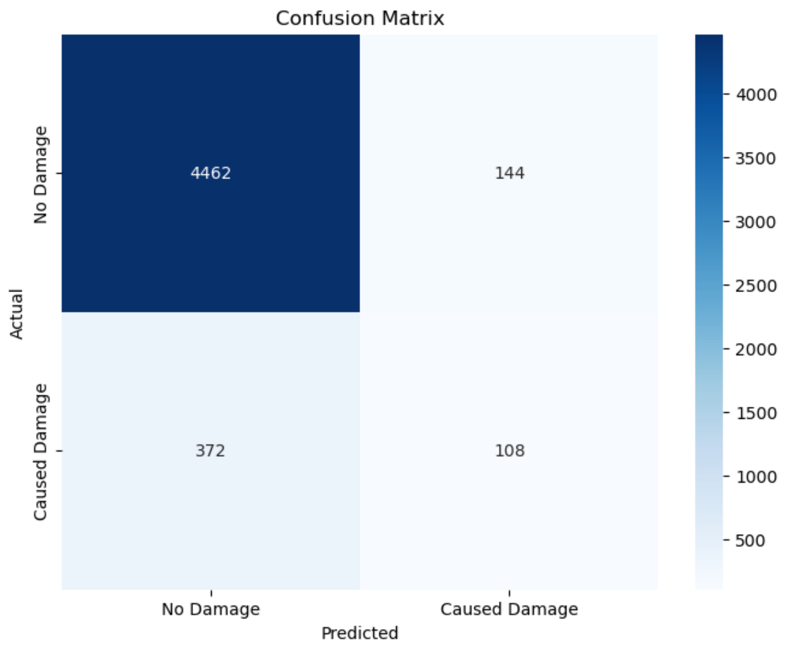
Evaluation Metrics

Accuracy and F1 Score:

For the majority class (0), the model performs well with precision (0.92), recall (0.97), and F1-score (0.95), reflecting its ability to correctly identify negatives. For the minority class (1), the performance is much weaker, with a precision of 0.43, recall of 0.23, and F1-score of 0.30, indicating significant challenges in identifying positive cases. The macro-average F1-score is 0.62, suggesting poor overall balance between classes, while the weighted-average F1-score of 0.88 reflects the model's bias toward the majority class due to the class imbalance.

```
Accuracy: 0.90
ROC AUC: 0.60
f1 score: 0.30

Confusion Matrix:
[[4462  144]
 [ 372  108]]

Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.97      0.95      4606
           1       0.43      0.23      0.30       480

    accuracy                           0.90      5086
   macro avg       0.68      0.60      0.62      5086
weighted avg       0.88      0.90      0.88      5086
```

Confusion Matrics:



The sensitivity and specificity are respectively: The model performs well in predicting "No Damage," as evidenced by the high number of true negatives (4462) and relatively low false positives (144). However, the model struggles with identifying "Caused Damage," as indicated by the high number of false negatives (372) compared to true positives (108). This suggests the model has low sensitivity for the minority class.

$$\text{specificity} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} = \frac{4462}{4462 + 144} = 0.968736$$

$$\text{sensitivity} = \frac{\text{True Negative}}{\text{True Negative} + \text{False Positive}} = \frac{108}{108 + 372} = 0.225$$

Conclusion and Future Directions

The classification model achieves an overall accuracy of 90%, reflecting strong performance in predicting the majority class ("No Damage"). The high specificity of 0.97 highlights the model's effectiveness in correctly identifying negative cases, as evidenced by the confusion matrix, where 4462 true negatives were correctly classified compared to 144 false positives. However, the model struggles to identify the minority class ("Caused Damage"), with a recall (sensitivity) of only 0.23 and an F1 score of 0.30, indicating significant difficulty in detecting rare events. Furthermore, the ROC AUC score of 0.60 reflects limited ability to discriminate between the two classes. While the macro-average F1 score of 0.62 underscores the imbalance in the model's performance across classes, the weighted F1 score of 0.88 is disproportionately influenced by the dominant class.

To improve the model's sensitivity for the "Caused Damage" class, future work should focus on addressing class imbalance through techniques like oversampling the minority class (e.g., SMOTE) or undersampling the majority class. Additionally, incorporating class-weighted loss functions could penalize misclassifications of the minority class and improve recall without significantly impacting overall accuracy.

References

[1] Tapendu, Bird Strike by Aircrafts Dataset. Available at: https://www.kaggle.com/datasets/iamtapendu/bird-strike-by-aircafts-data/data

[2] Koehrsen, W., Bayesian Logistic Regression in Python. Available at: https://github.com/WillKoehrsen/Data-Analysis/blob/master/bayesian$_{log_r}eg/Bayesian - Logistic - Regression.ipynb$

[3] Reding, A., Bayesian Logistic Regression: Arrhythmia Example. Available at: https://areding.github.io/6420-pymc/unit7/Unit7-arrhythmia.html

[4] Goldinlocks, Bayesian Logistic Regression with PyMC3. Available at: https://goldinlocks.github.io/Bayesian-logistic-regression-with-pymc3/

[5] How to Use Bayesian Inference for Predictions in Python. Available at: https://towardsdatascience.com/how-to-use-bayesian-inference-for-predictions-in-python-4de5d0bc84f3

[6] 8 Distributions Every Data Scientist Should Know. Available at: https://www.persuasivepython.com/8-distributions

[7] Building a Bayesian Logistic Regression with Python and PyMC3. Available at: https://towardsdatascience.com/building-a-bayesian-logistic-regression-with-python-and-pymc3-4dd463bbb16