Comparing Random Forest and Gradient Boosted Decision Tree Models in Sea Turtle Nesting Classification

Audrey Moore (Advisors: Dr. Pu, Dr. Schmutz, Dr. Seals)

2026-03-29

Background, Motivation, and Introduction

This research seeks to examine geomorphological features, such as foreshore slope and beach slope, as predictors for sea turtle nesting. These relationships are not always linear - there is some optimal level of slope or distance that may prevent flooding without overexerting the turtle.

Random forests and gradient boosted decision trees are common machine learning approaches for modeling non-linear relationships. While both have become a popular approach in classifying complex ecological outcomes for their predictive power, Random Forest models are more common.

Research questions:

- Which model performs better on classifying nesting sites?
- Which variables are most meaningful in classifying nesting sites?
- Does performance degrade when excluding certain variables?

Models will be evaluated on accuracy, precision, recall, F1-score, and AUC.

Methods

RF and GBDT classification models are both ensemble methods, however, they differ in learning techniques. RF and GBDT models may be prone to over fitting and difficult to interpret. Gradient boosting is sensitive to hyperparameters, but often performs better than RF when tuned correctly.

Random Forest

- Trees trained independently and results averaged to reduce overfitting
- n_estimators
  - Determines the number of trees in the ensemble
  - Larger is better - “Build as man as you have time/memory for”
- max_features
  - Determines the number of features selected for each tree
  - $\sqrt{\text{number of features}}$ for classification models

GBDT

- Trees trained sequentially to improve at each iteration
- n_estimators
  - Determines the number of trees in the ensemble
  - Increasing may lead to overfitting
- learning_rate
  - Determines strength of correction from tree to tree
  - Lower learning rate requires higher number of trees

Code Sources and Implementation

Sources:

- Code for this project sourced from previous coursework (STA6210), Machine Learning with PyTorch and Scikit-Learn, and Introduction to Machine Learning with Python.
- Code refinement and troubleshooting performed using ChatGPT.

Implementation:

- Visuals and exploratory analysis performed in R
- Machine learning and analysis performed in Python

Exploratory Analysis

Data | Missingness | Proportion Nested | Box Plots | Correlations

Data Element	Description
Nested (outcome)	Nesting status (0=did not nest; 1=did nest)
Foreshore Slope	Slope angle of the wet beach (meters)
Beach Slope	Slope angle of the dry beach (meters)
Nest Elevation	Mean sea level elevation of the nest (meters)
Dune Height	Mean sea level elevation of highest the dune feature at the nest location (meters)
Nest Distance	Crawl distance (meters)
Nest Year	Year of data collection (2010, 2016, 2020)
Nest Name	Unique ID for each nest

Data Preparation

Feature/Target Split | Train/Test Split | Imputation | Scaling

```
1  import pandas as pd
2  import numpy as np
3  import openpyxl
4
5  df = pd.read_excel('1-data/1-1 Ratio_Sea Turtle data_2010-2016-2020.xlsx')
6  X = df[['NestDist', 'NestElev', 'DuneHeight', 'BeachSlope', 'ForeshoreSlope', 'Nestyear']].values
7  y = df.Nested
8
9  print('Class labels:', np.unique(y))
```

```
Class labels: [0 1]
```

Random Forest Model

Building & Fitting | Optimizing Threshold | Feature Importance

```
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier(n_estimators=1000,
                                  max_features = "sqrt",
                                  random_state=1989)

rf_model.fit(X_train_std, y_train);

rf_pred = rf_model.predict_proba(X_test_std)[:, 1]
```

Gradient Boosted Decision Trees

Building & Fitting | Optimizing Threshold | Feature Importance

```
import xgboost as xgb

xgb_model = xgb.XGBClassifier(objective='binary:logistic',
                              n_estimators = 500,
                              learning_rate = 0.1,
                                  random_state=1989)

xgb_model.fit(X_train_imp, y_train);

xgb_pred = xgb_model.predict_proba(X_test_imp)[:, 1]
```

Full Model

Random Forest

```
Accuracy: 0.5877192982456141

Precision: 0.5656565656565656

Recall: 0.9333333333333333

F1-score: 0.7044025157232704

ROC-AUC: 0.6217592592592592

Confusion Matrix:
 [[11 43]
 [ 4 56]]
```

GBDT

```
Accuracy: 0.5964912280701754

Precision: 0.57

Recall: 0.95

F1-score: 0.7125

ROC-AUC: 0.5734567901234567

Confusion Matrix:
 [[11 43]
 [ 3 57]]
```

We see that both models performed about the same. GBDT slightly higher on all measures other than ROC-AUC.

Reduced Model

The reduced model does not include dune height or foreshore slope.

Random Forest

```
Accuracy: 0.5964912280701754

Precision: 0.57

Recall: 0.95

F1-score: 0.7125

ROC-AUC: 0.6646604938271604

Confusion Matrix:
 [[11 43]
 [ 3 57]]
```

GBDT

```
Accuracy: 0.5526315789473685

Precision: 0.5436893203883495

Recall: 0.9333333333333333

F1-score: 0.6871165644171779

ROC-AUC: 0.6382716049382715

Confusion Matrix:
 [[ 7 47]
 [ 4 56]]
```

The reduced RF performs about the same as the full model. The reduced GBDT slightly degrades in performance compared to the full model.

Conclusion

- Which model performs better on classifying nesting sites?
  - The GBDT model performed slightly better on the full model
  - The RF model performed better on the reduced model
- Which variables are most meaningful in classifying nesting sites?
  - Nest Elevation
  - Beach Slope
  - Nest Distance
  - Nest Year
- Does performance degrade when excluding certain variables?
  - Model performance improves for RF
  - Model performance degrades for GBDT