I. Background and Motivation

Wildlife trade is a multibillion dollar industry that simultaneously threatens the survival of many species and impacts the economic well being of large portions of the world.[1,2] In recent years the volume of wildlife trade, both legitimate and illicit, has increased at rates that have challenged customs and law enforcement officer's abilities to keep up with inspection requirements.[3] These inspections are critical to the detection and prevention of illicit wildlife trade, but the current methodology relies on a combination of individual agent's experience and physical inspections.

We believe that machine learning can be used to develop a binary classification system that automatically flags shipments with a high probability of being illicit so that customs agents can do targeted inspections on them. This will ideally create a two-step human-in-the-loop system that allows customs and enforcement officers to better utilize their limited resources by making informed decisions.

II. Project Overview

We aimed to answer the following questions in this project:

- Can we predict which wildlife import and export shipments are illicit?
- What features in imports and exports are important in predicting illicit wildlife shipments?
- Are there any differences in feature importance for illicit wildlife import vs. export shipments?

III. Data Source

Data Overview: In this project, we leveraged the Fish and Wildlife Service's (USFWS) Law Enforcement Management Information System (LEMIS) dataset[4] to build a prediction model that will accurately classify clear vs. seized wildlife shipments for the US customs agents. The LEMIS dataset includes labeled data on 5 years of wildlife imports and exports and their derived products into the United States, collected by the United States Fish and Wildlife Services (USFWS). The volume of records is large, including >2 million wildlife or wildlife product shipments. The dataset includes 24 variables and some of the variables have missing values or are recorded as unknown. We obtained the 2016 - 2020 LEMIS dataset through Freedom of Information Act (FOIA) requests. The FOIA was filed by a member of Professor Neil Carter's lab, the details of which are listed below.

Location	Google Drive
Format	A collection of quarterly .xlsx files
Variables	Product variables: species code, genus, species, subspecies, specific name, generic name, wildlife product description, wildlife product category, source Shipment variables: Control number, cartons, quantity, unit, reported value, country of origin, country of import/export, purpose, mode of transportation, shipment date,

[1] Andersson, Astrid Alexandra, et al. "CITES and beyond: Illuminating 20 years of global, legal wildlife trade." Global Ecology and Conservation 26 (2021): e01455.
[2] UNODC, World Wildlife Crime Report 2020: Trafficking in Protected Species.
[3] Scheffers, Brett R., et al. "Global wildlife trade across the tree of life." Science 366.6461 (2019): 71-76.
[4] Eskew, Evan A., et al. "United States wildlife and wildlife product imports from 2000–2014." Scientific Data 7.1 (2020): 1-8.

	shipment type (import/export) Outcome variables: action, disposition, disposition date
Records	2,093,505 records
Time periods	From 2016Q1 to 2020Q2[5]

Data Cleaning[6]: Since the LEMIS data was provided in multiple files, we first had to combine them with the unified column names. Once we combined all the datasets from 2016Q1 to 2020Q2, there were approximately 2M records of which we dropped ~200K duplicated records. Although the LEMIS dataset was already partially cleaned and organized, we still had to conduct general cleaning steps and evaluate each variable (e.g., check for distribution, outliers, missing values, etc.). In addition, the following steps were taken to further clean the dataset:

- Converted *shipment_date* from string to datetime variable
- Updated incorrectly recorded units
    - Example: Although "NO" was the correct unit, there were multiple records with "N0" as the unit
- Unified units and updated quantity accordingly
    - Example: Shipments measured in KG, GM, LB, and MG were all unified to KG
- Capitalized all country codes to keep them consistent
- Updated null values to specify the origin (column name)

Feature Creation: Based on the UNODC report there can be seasonality in imports/exports for specific species. In order to evaluate this further we created a new feature (*ship_date_mm*) that represents the shipment month.

Data Preparation: The LEMIS dataset included records of both wildlife imports and exports. Based on expert feedback indicating differences in both the data and relative importance, we wanted to train two separate models that will each predict illicit import vs. export shipments. After separating the data and the other cleaning steps there were a total of 1,889,845 records[7]: 299,340 (16%) export records and 1,590,505 (84%) import records.

Encoding: Through the EDA process, we found that most of the categorical variables had extremely high cardinality. For example, *species_code* alone had 14,055 unique values. Considering that most machine learning libraries require numerical inputs, it was essential we convert the categorical variables accordingly. One of the first approaches that we tried was OneHotEncoding. We quickly found that this approach expanded the feature space and created enormously sparse data frames, requiring much more computational resources than we had on hand.[8] After a few failed attempts and discussion with our project coach, we pivoted to research alternative ways to encode categorical variables. We considered

[5] Note that 2019 Q3 is missing from the FOIA results. For our analysis, we proceeded with the data as is.
[6] All code for this project can be found here: https://github.com/YinterestingProjects/milestone_2
[7] There were 287 records out of the total of 1,890,132 records that were marked as "T" and "*", which are neither imports or export records, and were not included in the analysis.
[8] The export dataframe required over 119GB just to load into memory after OneHotEncoding.

both ordinal encoding and target encoding. Given the limited time frame, we landed on the latter due to the fact that there is no inherent order within categorical variables like *species_code*, *generic_name*, or *genus*. Ordinal encoding could introduce additional bias from the random assignment of order. Target encoding, on the other hand, is a Bayesian approach that converts each category with the categorical variable to a numerical number based on its contribution to the predicted outcome.[9,10] The result is therefore much more meaningful for this data.[11]

IV. Unsupervised Learning

Originally we planned to leverage principal component analysis (PCA) as a necessary dimensionality reduction technique following OneHotEncoding of high-cardinality features. Pivoting to target encoding proved effective at eliminating the curse of dimensionality, but we still ran PCA to check if the feature set can be further compressed while retaining the variance of the dataset before supervised learning.[12]

Workflow steps:

- Import and export records were separated from the cleaned dataset
- Imports and exports were each divided into train-test sets (75/25) through a stratified split[13]
- Each X_train had a total of 16 features (14 categorical variables and 2 numerical variables), of which
    - target encoding[14] was performed on all categorical variables
    - standard scaling was performed post encoding on all variables
- PCA fitted and transformed each X_train from import and exports

Challenges and Solutions: We struggled with successful PCA implementation specifically around setting up and keeping track of the different pre-processing needs across the different column types. With target encoding on the majority of variables, the units of their numerical representation are the same and all values are bounded between 0 to 1. However, our numerical columns were on different scales and neither column types were centered on 0 with unit variance. Keeping in mind that our results could be part of a chained pipeline where training data folds need to be independently scaled by column data type, normalized accordingly, and reduced on dimensionality before fitting of a classifier, we faced the challenge of keeping track of these steps consistently. We found our solution in two scikit-learn methods: ColumnTransformer and Pipelines. ColumTransformer proved extremely useful for selectively transforming columns based on specified criteria, while Pipelines helped streamline chaining of different steps and avoid data leakage through inappropriate scaling and encoding.

Parameter Tuning: After successful implementation of PCA, we reviewed variance plots and scree plots to check how well PCA components captured variation in our dataset and determine the top *n_components* to keep.

---

[9] Cerda, Patricio, and Gaël Varoquaux. "Encoding high-cardinality string categorical variables." IEEE Transactions on Knowledge and Data Engineering (2020).

[10] Larionov, Michael. "Sampling Techniques in Bayesian Target Encoding." arXiv preprint arXiv:2006.01317 (2020).

[11] Note that the use of target outcome during encoding can cause overfitting, however, based on our results this was not an issue.

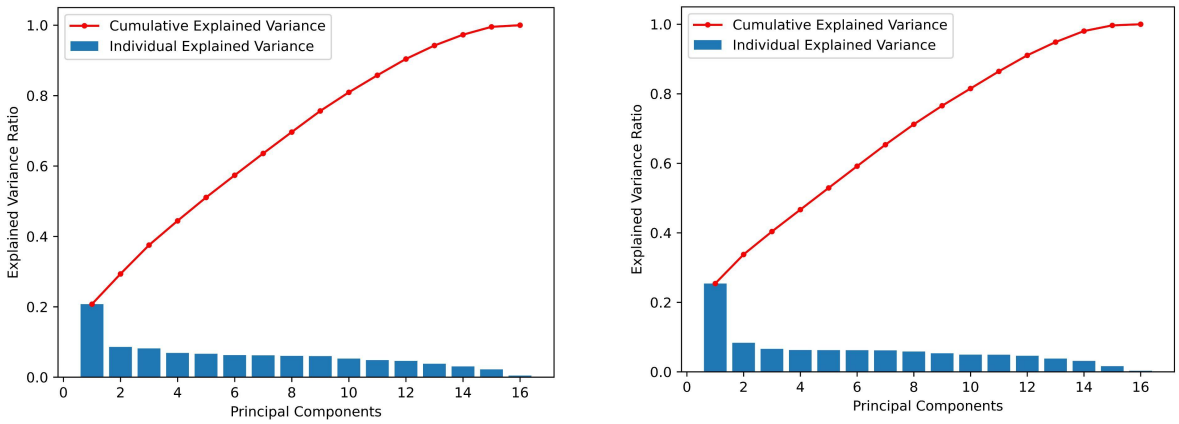
[12] Note the results of PCA were not used for downstream supervised learning due to time constraints. Our supervised learning feature selection was guided in part by subject matter expert input. This is a future direction we hope to explore.

[13] Due to the roughly 100:1 target class imbalance in our dataset, stratified splits were used to preserve target class ratios in train test sets.

[14] Micci-Barreca, Daniele. "A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems." ACM SIGKDD Explorations Newsletter 3.1 (2001): 27-32.

Figure 1. pareto plots of variance explained ratio for principal components

a. Exports b. Imports

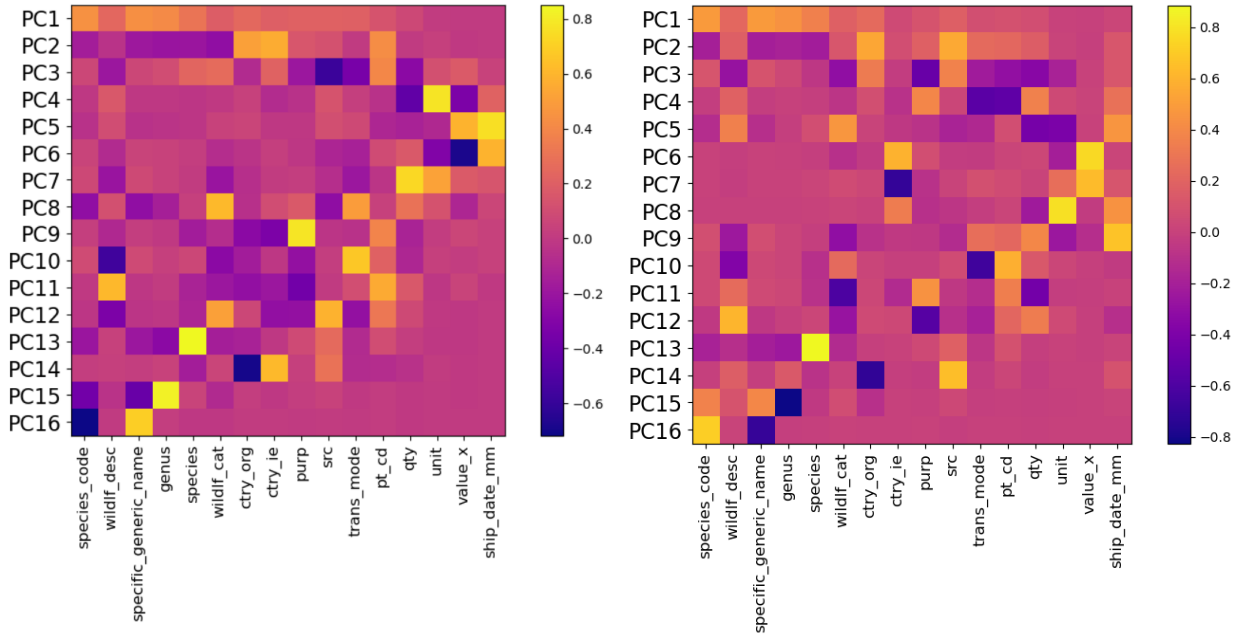


Variance plots (Figure 1) showed that the top 8 principal components across imports and exports captured 80% of the variation in our dataset. The first principal components contributed roughly 20% of the variance, and each of the remaining components contributed less than 10% of the variance. Our import and export scree plots (Appendix B.1) both had one steep drop followed by very long tails. Using the elbow heuristic, we arrived at similar results.

Discussion:

Figure 2. Correlation matrix plots

a. Exports b. Imports



We further analyzed what feature groupings each component is associated with via heat map for each component (Figure 2). From the first principal component, we found that *species_code, specific_generic_name, genus*, and *species* are covariants together across both imports and exports. This makes sense since species and genus descriptors are hierarchical in definition, species and species

encodes are mapped encodings, and generic name correlates closely with species. We also see that most of the other features are neutral in contribution, signifying that wild life description fields as listed earlier are the major contributors to the first 20% of variance within the dataset. The second component has a variation of positive and negative values. In particular, we see that from imports, country of wildlife origin (*ctry_org*) is correlated with source (*src*). And from exports, country of wildlife origin (*ctry_org*) is correlated with country of export (*ctry_ie*) and port of shipment (*pt_cd*). This difference between exports and imports is interesting to consider and matches subject matter expert understanding that export and import custom checks may differ in nature. Other notable findings include that *unit* and quantity (*qty*) were negatively correlated in exports, suggesting that more frequent units were associated with smaller quantities. This also made sense since there are laws regarding wildlife shipment packaging especially towards live specimens where the number of wildlife within the same shipment carton is limited for humane transport conditions. In imports, value and country of import (*ctry_ie)* is also closely correlated, this is an interesting find that hints at potential trends of high value items importing routinely into the U.S. from certain countries.

Dealing with the curse of dimensionality due to the large number of categorical features with high cardinality in our dataset presented a challenge in preparing the dataset for supervised learning. Although we were able to resolve this through more memory efficient encoding via target encoding, PCA results from above showed us that many of the categorical variables are correlated. These results are valuable in guiding further feature engineering. Given time constraints and our first iteration of tree-models having fewer requirements on preprocessing, we chose to prioritize our time on hyperparameter tuning.

As we head into Capstone where we hope to leverage more complicated neural network models, we plan to focus on more aggressive feature engineering. Some solutions that we intend to explore are

1) Elapsing high cardinality features such as species into more meaningful indicators via external datasets (ex. endangered level listing via USFWS Species Report)[15],
2) Leverage alternative PCA methods such as kernel PCA which better handles non-linearity and may capture most of the variations within two to three principle components,
3) Better resolution of multicollinearity between wildlife descriptor features through feature extraction.

V. Supervised Learning

Methods & Evaluation: During the previously discussed challenges confronted during PCA implementation, we proceeded with supervised learning through an expert-informed feature selection process to minimize multicollinearity between columns and select the smallest subset of features that would be most representative of the decision indicators used in manual inspections. The following 5 features (*species, genus, specific_generic_name, unit, qty*) were identified as containing overlapping or superfluous information solemnly used in decision-making. The remaining 11 features were used moving forward into model training and hyperparameter tuning .[16]

[15] As an example, *species_code* alone had 14,055 unique values and by leveraging external datasets, we may be able to create more meaningful groupings.

[16] See Appendix A for additional details on all of the selected, added, and dropped features.

Import and export data were separated and stratified[17] with a 75-25 train-test split on each dataset. Five-fold cross validation was used to examine model training stability, reduce overfitting, and obtain more statistically significant results from training. In order to avoid data leakage from encoding and scaling during cross validation, we built pipelines where each fold of training data independently follows the following steps:

- Target encoding on categorical columns
- Standard scaling on all column values to ensure zero mean and standard variance
- Imbalance handling via synthetic minority oversampling technique (SMOTE) [18]
- Fit transformation using classifier of choice

GridSearchCV was used to explore how different parameters affected key success metric performances. For tree-base models, we experimented with class_weight parameter inputs, pruning by max_depth, learning rates where applicable, and adjusting the number of estimators for ensemble methods. [19] For the regression model we experimented with class_weight, different solvers, and C regularization values.

Our evaluation metrics were decided through the consideration of our business goals and user context. Currently, wildlife shipments are manually flagged and then suspicious packages are sent through to specialized agents for a more thorough inspection before official release or seizure. Through conversation with end users and subject matter experts we determined that the ideal use case for a machine learning model would be to automate the shipment flagging and alert agents of any shipments deemed likely to be illicit. Since this is a two-step human-in-the-loop system, there is inherent tolerance to false positive suspicious shipments. Because of this, we chose to use recall as our primary success metric. Knowing the trade-off between recall and precision, we also leveraged F1 scores, the harmonic mean between recall and precision, as a secondary success metric.

Model Selection: While we included logistic regression as a baseline performance comparison model, our classifiers of choice for this project were primarily tree-based models for four reasons.

1) We are working with non-technical stakeholders, and even our subject matter experts are not overly familiar with machine learning. As a result we wanted to prioritize explainability and transparency which is a hallmark of tree based algorithms.
2) Unlike a lot of other models, tree based models generally handle categorical data without extensive manipulation beyond some form of numerical encoding.[20]Additionally tree-based models are non-parametric and benefit from the less strict assumptions required by alternative classification models like Logistic Regression.
3) Tree based models have easily accessed and interpreted feature importances which is one of our primary questions in this project.
4) We intend to continue this project into Capstone with more complex models such as neural networks and wanted to first explore simpler models to better inform decisions in a more “black box” environment.

[17] Stratification allowed both the train and test set to maintain the same ratio of target classes

[18] Due to the large imbalance observed between the binary target class of cleared shipments and seized shipments (approximately 100:1 ratio), we experimented with imbalance handling via SMOTE from the imblearn library. A control group pipeline was set up with no SMOTE application and a treatment group pipeline with SMOTE. Given more time we would like to have experimented with multiple upsampling percentages as well as other techniques, but the model training time combined with the limited project time precluded this.

[19] See Chart 1 for final tuned model parameters.

[20] Some models are attempting to navigate the difficulty of using string encoded categorical features but we did not have much luck with implementing the current versions.

Performance: We began model selection and tuning with 6 different types of models[21] which we ran in default configurations across both import and export datasets. This established a baseline at an average recall rate below .45 on the export data and .65 on the import data across all models. From this point all models were fed into the previously described pipeline resulting in 12 models going through GridSearchCV.[22] Due to time and computational constraints, 8 of the models completed training. Out-of-sample recall and F1 performance scores for best estimators of completed models are reported in Figure 3.[23] Details around in-sample recall rates and best parameters from each of the estimators are listed in Appendix C.1.

Figure 3. Summary Bar Chart for GridSearchCV Best Estimators Recall and F1 Performance

All models, with the single exception of the export LGBM (SMOTE), finished with out-of-sample[24] recall rates between .55 and .83 post hyperparameter tuning with GridSearchCV. These results represented significant recall improvement from the default configurations. However, it's evident that our decision to prioritize recall came at a significant cost in terms of F1 score and overall precision across almost all models. The notable exception to this trend can be seen in the performance of the LightGBM (SMOTE) model which had a final F1 score of .74 and a recall of .62.

Our imbalance handling efforts via SMOTE proved to have had limited improvement on key evaluation metrics. On average, SMOTE classifier pipelines performed on par if not worst in recall and F1 score across different estimators. With the exception of the import LightGBM (SMOTE) pipeline, where the addition of SMOTE drastically improved F1 score while sacrificing some of recall. This is a noteworthy pattern that calls for more experimentation. Specifically, our results can potentially benefit from iterating on different ratios of minority class oversampling as opposed to the single 1:1 minority / majority ratio used due to time constraints.

[21] Scikit Learn was used for Logistic Regression, Decision Tree, and Random Forest. LightGBM, XGBoost, and Fast Interpretable Greedy-Sum (FIGS) all came from their respective libraries. This count also does not include the dummy classifier.

[22] Several of the libraries were completely new to the team and come with a large number of tuning hyperparameters. As a result, and in conjunction with 5 fold cross-validation for each combination of hyperparameters, more complex models ran over 19,000 gridsearch iterations. Using the results from those runs enabled faster follow-on hyperparameter tuning by reducing the number of hyperparameters focused on or the value ranges.

[23] XGBoost, XGBoost (SMOTE), FIGS, and FIGS(SMOTE) all took several days to run through gridsearch on the import data, even after reducing the number of hyperparameters considered. These 4 models will be moved to Capstone for reporting.

[24] Tested on hold-out test set

Worth noting, the respective sizes of the available data across imports and exports also appear to have played an important role in model performance. From start to finish, the tree based models of all varieties generally had a 10% performance increase in recall on the import data vs the export data, and F1 performance was between 2-3 times higher[25] for the import data vs the export data.

Hyperparameter Sensitivity: Aside from differences in imbalance handling via SMOTE and export and import data volume differences, we also discovered the following during hyperparameter tuning:

Logistic Regression
experimented with class_weight, different solvers, and C regularization values

- model performance was extremely sensitive to the class_weight parameter without SMOTE, on average recall increased from 0.2 to 0.5 when class_weight was set to *balanced* compared to other customized weighting of the binary target class
- both C regularization and solver changes did not drastically affect recall

Decision Tree
experimented with class_weight parameter inputs, and pruning by max_depth

- model performance was sensitive to the class_weight parameter without SMOTE, on average recall increased from 0.4 to 0.6 when class_weight was set to *balanced* compared to other customized weighting of the binary target class.
- model performance was also sensitive to the max_depth parameter, especially as it paired with different class_weight choices

Random Forest
experimented with class_weight parameter inputs, pruning by max_depth, learning rates where applicable, and adjusting the number of estimators for ensemble methods. [26]

- model performance was sensitive to the class_weight parameter without SMOTE, on average recall increased from 0.3 to 0.6 when class_weight was set to *balanced* compared to other customized weighting of the binary target class.
- model performance was to a less degrees also sensitive to the tree max_depth parameter and number of trees used in the ensemble (n_estimator), especially as it paired with different class_weight choices

LightGBM
experimented with n_estimators, limiting max_depth, limiting num_leaves, different learning_rate values, and both is_unbalance values.

- To tune LightGBM we experimented with the number of boosted trees (n_estimators), the max depth trees could grow to (max_depth), the number of leaf nodes a tree can have (num_leaves), the learning rate (learning_rate), and whether or not the dataset should be treated as balanced (is_unbalance).
- LightGBM was generally not very sensitive to most of the parameters, achieving recall scores within a range of ±.05 regardless of using wide ranges of boosted trees, tree depth and leaf limitations, and the learning rate. The exception to this was the is_unbalance parameter, which had a significant impact on the model performance.

[25] LightGBM(SMOTE) had almost a 20% performance boost in recall, and a 5x increase in F1 performance between the datasets.
[26] See Chart 1 for final tuned model parameters.

Feature Importance

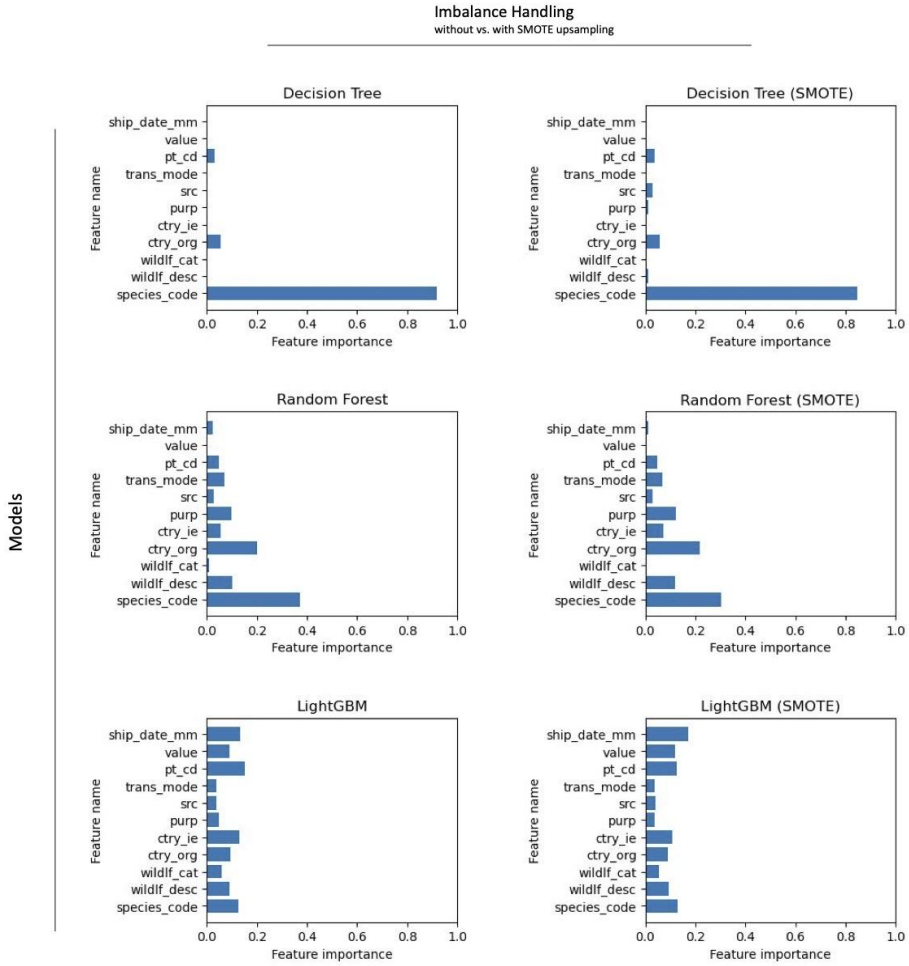
In order to address the remaining objective questions:

- What features in imports and exports are important in predicting illicit wildlife shipments?
- Are there any differences in feature importance for illicit wildlife import vs. export shipments?

we zeroed in on the best tree-based estimators from our GridSearchCV results.

Using the *feature_importance_* output property, we compared how much estimators relied on each input feature to make predictions on illicit wildlife shipments. Comparison outputs for import models are shown in Figure 4 and export models are available in Appendix C.2.

Figure 4. Import Models Feature Importance Comparison
Across 5-fold GridSearchCV Best Estimators (Seized = 1, Clear = 0)



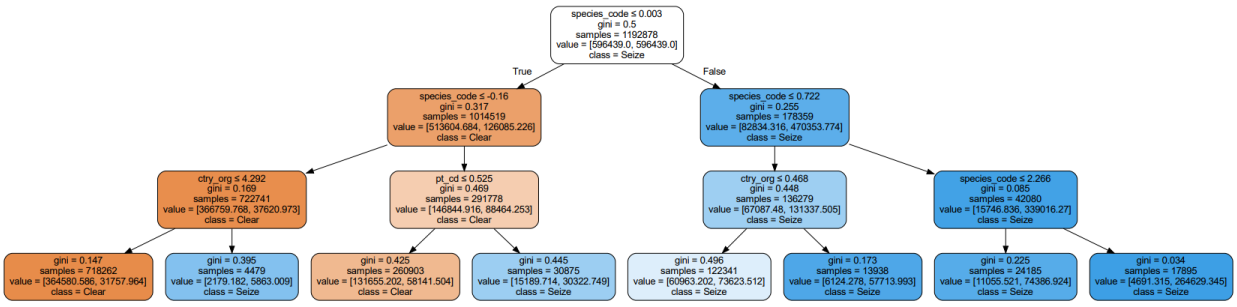
In general, we observed that imbalance handling via SMOTE did not have a significant impact on feature importance across all models in export and import. Feature importance was model specific, but import and exports shared some similarities among the best estimators within the same model family. For example, both import and export decision tree models heavily relied on *species_code* (0.6 to 0.9 in feature importance). LightGBM feature importances for both import and export showed more leveled feature importance, with all features contributing importance below 0.2. *Species_code*, country of

import/export (*ctry_ie*), and port of shipment (*pt_cd*) were among the top contributing features for both import and exports. Given that the LightGBM model at its highest recall rate also had the best F1 score among all models, these features could be key to reducing false positives. Random Forest feature importance differed between import and export models. Import models favored *species_code*, wildlife description (*wildlif_des*), country of origin (*ctry_org*) and purpose of shipment (*purp*). Export models favored species_code, country of origin (*ctry_org*), port of shipment (*pt_cd*) and transportation mode (*trans_mode*). These similarities and differences hint at the possibility that the decision making process for seizures in both export and import may be more alike than what we imagined. As there exist a five-fold difference[27] between the volume of import and export data, this is something we hope to confirm further with more data in the Capstone.

Trade offs: There are two main trade-offs we can identify in our choices for supervised learning.

1) Precision vs Recall. Due to the human in the loop nature of this project, we deliberately chose to favor recall at the expense of precision. The best models we hoped would be able to maintain a high recall rate, without too much sacrifice on precision. As discussed in the performance section, this was a difficult balancing act and why we chose F1 scores as a secondary success metric.
2) Interpretability vs Accuracy/Performance. As previously discussed, we chose to prioritize interpretable models knowing in advance that they would likely suffer in comparison to more advanced models such as neural networks in terms of overall performance. Tree-based models especially benefit from user-friendly tree plots as shown in Figure 5.

Figure 5. Import GridSearchCV Best Decision Tree Estimator[28]



With these trade-offs in mind, the following models were selected as top performers for predicting illicit shipments in imports and exports[29]:

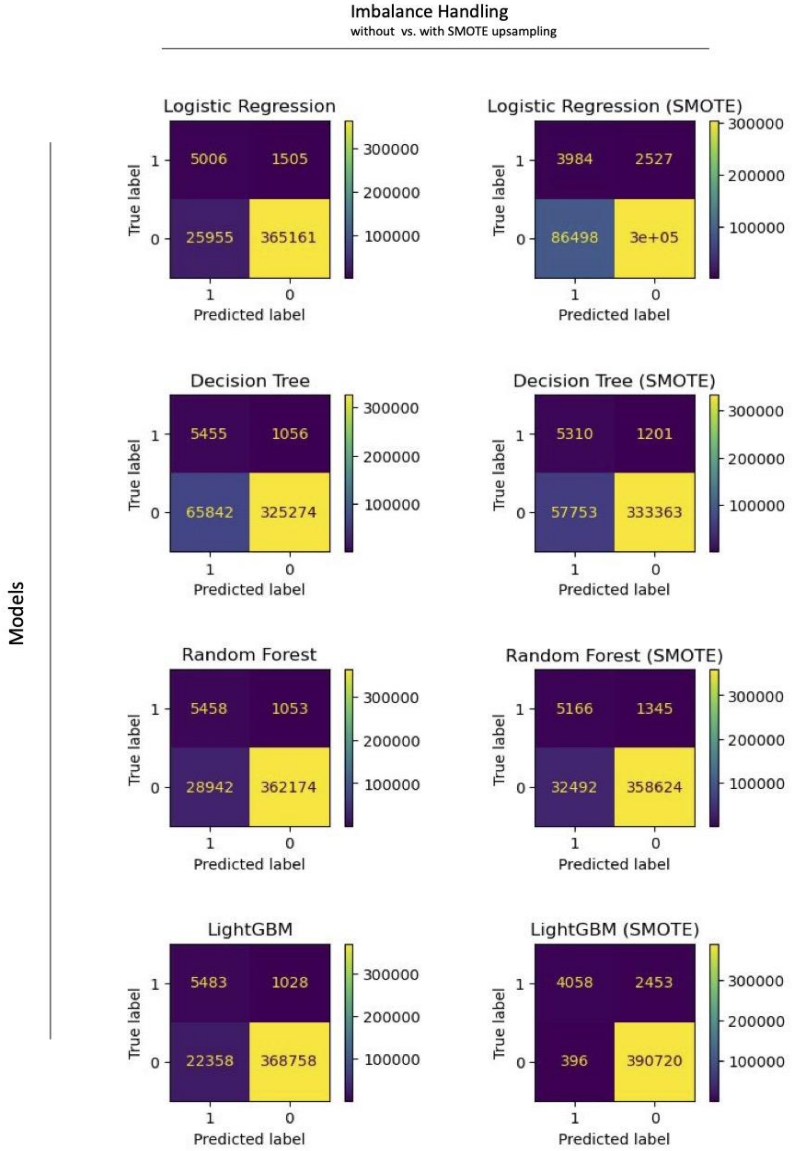
- Export: Decision Tree (recall: 0.823, F1: 0.073); Decision Tree (SMOTE) (recall: 0.745, F1: .057); Random Forest (recall: 0.728, F1: 0.074).
- Import: LightGBM (recall: 0.842, F1: 0.319); Random Forest (recall: 0.838, F1: 0.267); Decision Tree (recall: 0.728, F1: .074).

Failure analysis

[27] 299,340 (16%) export records and 1,590,505 (84%) import records
[28] export GridSearchCV Best DecisionTree estimator tree plot
[29] Note while each model managed to obtain a relatively high recall score, the highest F1 score achieved was only .319.

An overall failure analysis was first conducted on all gridsearch best estimators for each model family across import and export via confusion matrices. Comparison across different models on import out-of-sample prediction fails are shown in Figure 6. Comparison across export models are available in Appendix C.3.

Figure 6. Import Out-Of-Sample Prediction Failure Analysis Comparison
Across 5-fold GridSearchCV Best Estimators (Seized = 1, Clear = 0)



We found that due to the large imbalance of target classes in our dataset, our models collectively struggled on both recall and precision. While our hyperparameter tuning maximized recall score, many of the models faced the trade-off of higher counts of false positives. While this is tolerable within a human-in-loop design, we would still like to minimize human inspection if possible without sacrificing on

catching false negatives. Specifically, we conducted more in-depth failure analysis to understand the errors made by the best performing model in import and export. LightGBM was selected for imports. Decision tree was selected for exports.

For imports, LightGBM achieved the best recall and F1 score on balance. This estimator had an in-sample recall score of 0.829, and an out-of-sample predictive recall of 0.842 on the hold-out set. Among the predicted fails, we found that:

- Out of the 1,028 false negative errors, the most frequent errors (72; 7%) we observed came from shipments related to American Alligators
- Out of the 22,358 false positive errors, the most frequent errors (1,628; 7%) we observed also came from shipments related to American Alligators

For exports, Decision Tree had an in-sample recall score of 0.628, and out-of-sample predictive recall of 0.823 on the hold-out test set. Among the predicted fails, we found that:

- Out of the 114 false negative errors, the most frequent errors (13; 11%) we observed came from shipments related to American Alligators
- Out of the 13,351 false positive errors, the most frequent errors (824; 6%) we observed came from shipments related to White-tailed Deer; but shipments related to American Alligator (204; 1.5%) was also observed with false positive errors

Due to the fact that most of our predictive fails centered on American Alligators, we were curious to investigate further and better understand what could have contributed to these errors. When reviewing the species in context of the entire shipment, we found that American alligators are the second highest category in shipment volume in both import and export. Despite their large volume as a category, only 1.6% of alligator shipments were seized in imports (out of a total of 51,984) and 0.6% were seized in exports (out of a total of 17,861). Thinking back to our feature representation for species, target encoding on Alligators would have overwhelmingly favored this category as an indicator for cleared. This hints at additional layers of feature engineering and representation necessary to help our models better discern the difference between normal and illicit shipments.

Discussion

Through our attempt at training a binary classification model that predicts illicit wildlife shipments based on the LEMIS dataset, we found that we are able to predict illicit wildlife shipments to reasonably high recall after hyperparameter tuning on selected interpretable models. However, this came at the expense of precision, which in context will mean that manual inspection efforts will scale with the volume of shipments coming through. Although this is better than the current state of affairs on the ground where each shipment is hand-checked, we seek better balance in future iterations of our project in Capstone. We also found that import models fared much better on balance across both recall and F1 score with the larger volume of training data. We hope to extend the export model training to include more historical data in Capstone. We also discovered that feature importance varied among the best performing models, indicating that there's a bit of variability in decision boundaries. This makes sense in retrospect with what we observed from the principal component analysis where significant correlations exist in the dataset. Our findings here shall serve as guidance for the upcoming Capstone where we hope to build on these efforts and further refine our modeling approach.

Along the way we encountered several surprising results and difficulties, the most memorable of which was properly handling imbalance target classes to achieve reliable predictive results. We knew from our MADS learnings that accuracy would be a misleading evaluation metric with the significant imbalance of roughly 100:1 in our binary target, but we were properly challenged in finding suitable alternatives. Our experiments with class_weight parameters and up and down sampling techniques proved fruitful based on sensitivity analysis, but really only scratched the surface of the possibilities. We intend to double down on refining the approach to rebalancing the dataset in the near future.

Next Steps

We intend to continue this project into Capstone, which means there is still a significant amount of work to be done. The next steps we want to take can be broken down into three different primary categories.

1) Existing Work Analysis: We want to perform additional performance analysis on the existing model choices. One of the reasons we chose the tree-based models is their interpretability and relative simplicity, and we want to further leverage that by comparing classification decision thresholds if we make changes such as which score to prioritize and how much synthetic data we add into the training sets.
2) Additional Data Manipulation: We want to explore what happens when we add additional information such as socioeconomic or international trade routes to the LEMIS data.[30] [31] We also want to explore additional feature engineering with the existing data, potentially by expanding features instead of reducing like we did in this first approach.
3) More Advanced Models: This is a very complex dataset and it is possible we will get better results from more advanced models such as probabilistic modeling and neural networks.

VI. Ethical Considerations

In the LEMIS dataset, there is no information of the Customs and law enforcement nor any information on who may be sending or receiving these wildlife shipments, thus there are no re-identification issues.

Although only about 1.5% of the LEMIS dataset are seized wildlife shipments, our model currently predicts 1 out of 6 shipments to be illicit. If this model were to be implemented, it is possible that we may incorrectly label legal shipments and potentially increase more work for the Customs and law enforcement agents and delay the shipments.

In addition, there is a concern that the seized shipments are not a complete representation of all illicit shipments as this is largely dependent upon how well the Customs and law enforcement agents are trained to detect these shipments. Due to these biases, our model that is trained on this dataset could potentially be excellent at catching shipments that have been largely illicit, however, it may be unable to uncover other types of illicit shipments; and illicit wildlife traders may take advantage of these biased algorithms. In order to prevent these unintended negative consequences, we would likely have to randomly select labeled legal shipments for ongoing evaluation.

---

[30] **Socioeconomic data:** International Monetary Fund (IMF) World Bank's World Economic Outlook dataset to extract each country's gross domestic product (GDP). We believe that clustering import and export countries based on their socioeconomic data could provide additional insights and serve as an important feature.

[31] **Wildlife data mapping:** USFWS Species Report and CITES Trade Database. These external datasets will not only help us better understand the relationships between *species_code, genus, species, subspecies, specific_name, generic_name, wildlf_desc, and wildlf_cat*, but also wildlife species that are often exploited in international trades can serve as a feature.

VII. Statement of Work

We organized all of our background articles, data, data-related documents in Google Drive and all of our codes and analytic outputs in GitHub. We divided up equally in terms of building pipelines for data cleaning, preprocessing, supervised/unsupervised model building and evaluation.

- Jina Park: Background research, data cleaning, supervised, project report
- Joshua Haskins: Data ingestion, unsupervised, supervised, project report
- Sally Yin: EDA, data cleaning, unsupervised, supervised, project report

Appendix A Supplemental for Feature Engineering

A.1 Selected Features

Feature	Unique Count	Unknown Count	NaN Count	Data Type	Encoding
Species code (*species_code*)	14,055	243	0	categorical	Target Encoding
Wildlife description (*wildlf_desc*)	97	1	0	categorical	Target Encoding
Wildlife category (*wildlf_cat*)	25	0	1,736	categorical	Target Encoding
Country origin (*ctry_org*)	253	21,284	19,975	categorical	Target Encoding
Country import/export (*ctry_ie*)	248	221	21,453	categorical	Target Encoding
Purpose (*purp*)	13	164	0	categorical	Target Encoding
Source (*src*)	12	9,555	0	categorical	Target Encoding
Mode of transport (*trans_mode*)	9	883	0	categorical	Target Encoding
Port of shipment (*pt_cd*)	88	38	0	categorical	Target Encoding
Value of shipment (*value*)	86,604	157,186	0	float64	Target Encoding

A.2 Added Feature

Feature	Explanation
Month of shipment (*ship_date_mm*)	Derived from the date of shipment to evaluate seasonality effect

A.3 Dropped Features

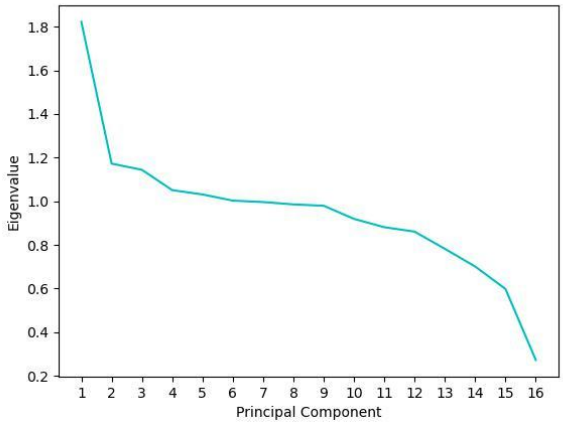
Feature	**Explanation**
Control number (*control_number*)	This is an ID number that gets assigned to each shipment; we have determined that this will not be appropriate as a feature
Subspecies (*subspecies)*	Approximately 97% of the subspecies were missing
Species* (*species*)	We dropped this feature as we decided to use species code as a feature
Genus* (*genus*)	In the hierarchy of biological classification, genus comes above species, thus we decided to keep species and drop genus as a feature
Specific name (*specific_name*)	Approximately 23% of the specific name was missing
Generic name (*generic_name*)	Generic name tended to be too broad and highly correlated with species and genus names
Specific/generic name* (*specific_generic_name*)	We initially created this feature by concatenating generic and specific names in order to test out a feature that had a bit more specificity to the generic name as it was a bit too broad; but due to high percentage of missing values from specific name, we determined that this will not be helpful as a feature
Unit* (*unit*)	There are a total of 13 different units (e.g., weight, volume, area, length) even after unifying units; due to these differences, we determined that this will not be helpful as a feature
Quantity* (*qty*)	Due to this quantity being associated with 13 different units, we determined that this will not be helpful as a feature
Cartons (*cartons*)	There are 4,571 unique values, ranging from -13 to 10,547,004; not only the distribution of this value was too wide but also there were no patterns associated with unit/quantity features as they are too many different units
Disposition date (*disp_date*)	This date won't be available as disposition date only gets assigned when the decision (cleared vs. seized) on the import has been made
Shipment date (*ship_date*)	There are a total of 1,461 unique shipment dates out of total of possible 1,825 unique possible dates (in 5 years); we have determined that this will not be appropriate as a feature

*5 features (species, genus, specific/generic name, unit, and quantity) were dropped after PCA

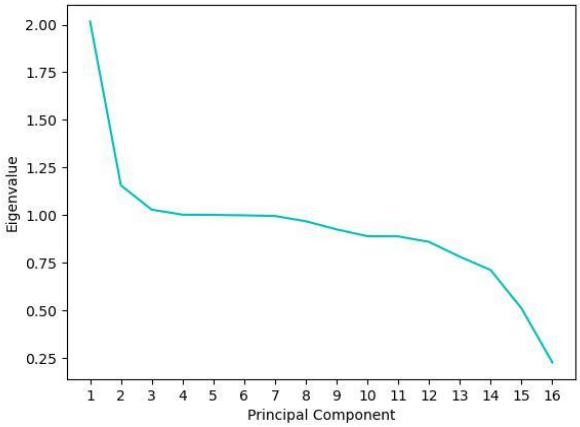
Appendix B Supplementals for Unsupervised Learning

B.1 Scree Plots

a. Exports



b. Imports



Appendix C Supplementals for Supervised Learning

C.1a GridSearchCV in-sample results based on maximizing recall and 5-fold cross validation

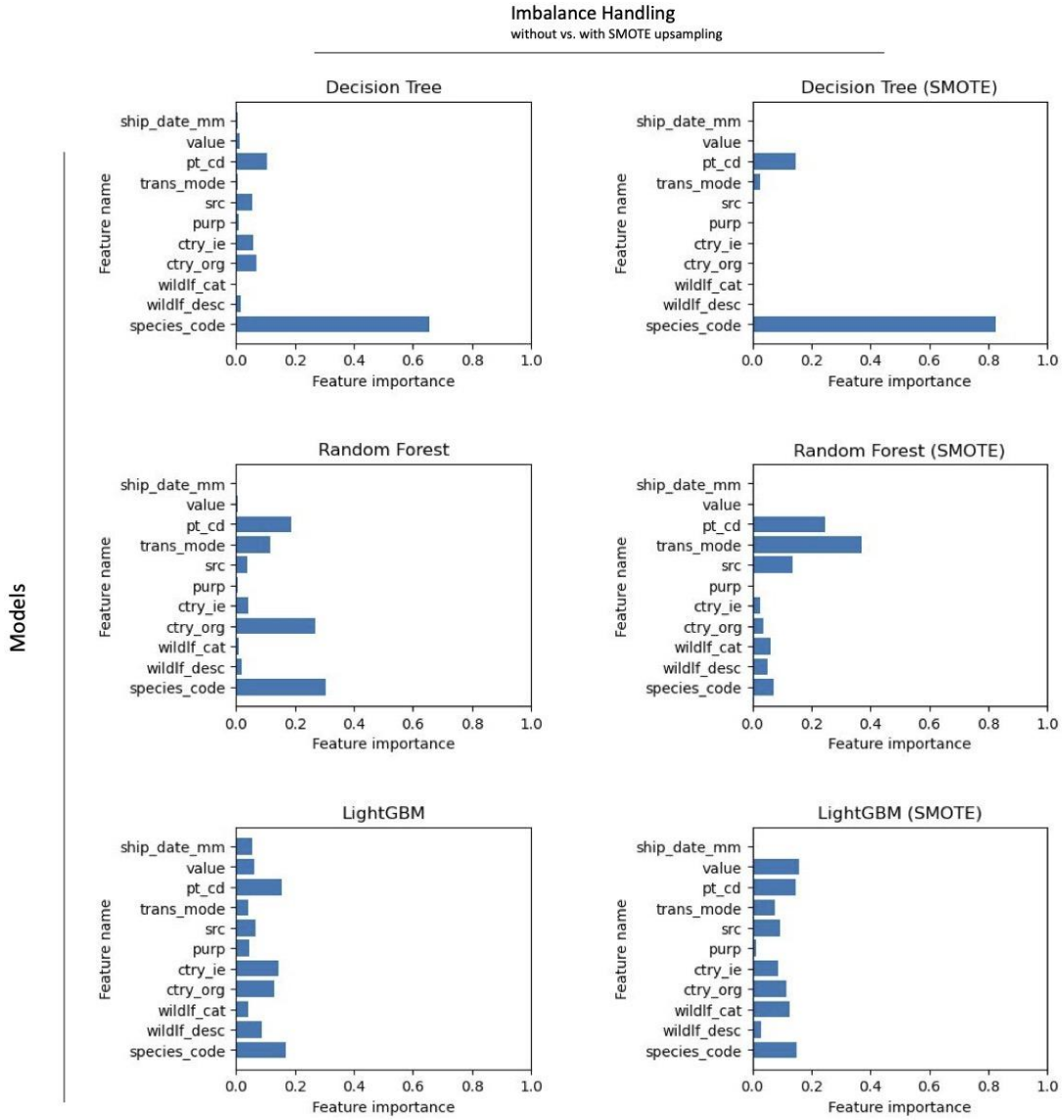
		Log Regression	Decision Tree	Random Forest	LightGBM
no imbalance handling	**Import Recall Rate**	0.7551	0.8241	0.7958	0.8290
	Import Best Param	'C': 0.1, 'class_weight': 'balanced', 'solver': 'lbfgs'	'class_weight': balanced, 'max_depth': 3	'class_weight': balanced, 'max_depth': 8, 'n_estimators': 50	'is_unbalance': True, 'learning_rate': .05, 'max_depth': 15, 'n_estimators': 1000, 'num_leaves': 20
	Export Recall Rate	0.5625	0.6284	0.6237	0.6492
	Export Best Param	'C': 0.1, 'class_weight': balanced, 'solver': 'lbfgs'	'class_weight': balanced, 'max_depth': 7	'class_weight': balanced, 'max_depth': 3, 'n_estimators': 20	'is_unbalance' : True, 'learning_rate': .05, 'max_depth': 10, 'n_estimators' : 100, 'num_leaves' : 10
SMOTE sampling	**Import Recall Rate**	0.7571	0.7684	0.7755	0.6166
	Import Best Param	'C': 0.5, 'class_weight': None, 'solver': 'lbfgs'	'class_weight': None, 'max_depth': 4	'class_weight': balanced, 'max_depth': 3, 'n_estimators': 50	'is_unbalance': False, 'learning_rate': .05, 'max_depth': -1, 'n_estimators': 1000, 'num_leaves: 70'
	Export Recall Rate	0.5495	0.5755	0.6362	0.4614
	Export Best Param	'C': 0.1, 'class_weight': None, 'solver': 'lbfgs'	'class_weight': None, 'max_depth': 3	'class_weight': balanced, 'max_depth': 3, 'n_estimators': 3	'is_unbalance': False, 'learning_rate': .05, 'max_depth': 40, 'n_estimators': 100, 'num_leaves': 10

C.1b GridSearchCV Best Estimators Out-of-Sample Recall and F1 Scores

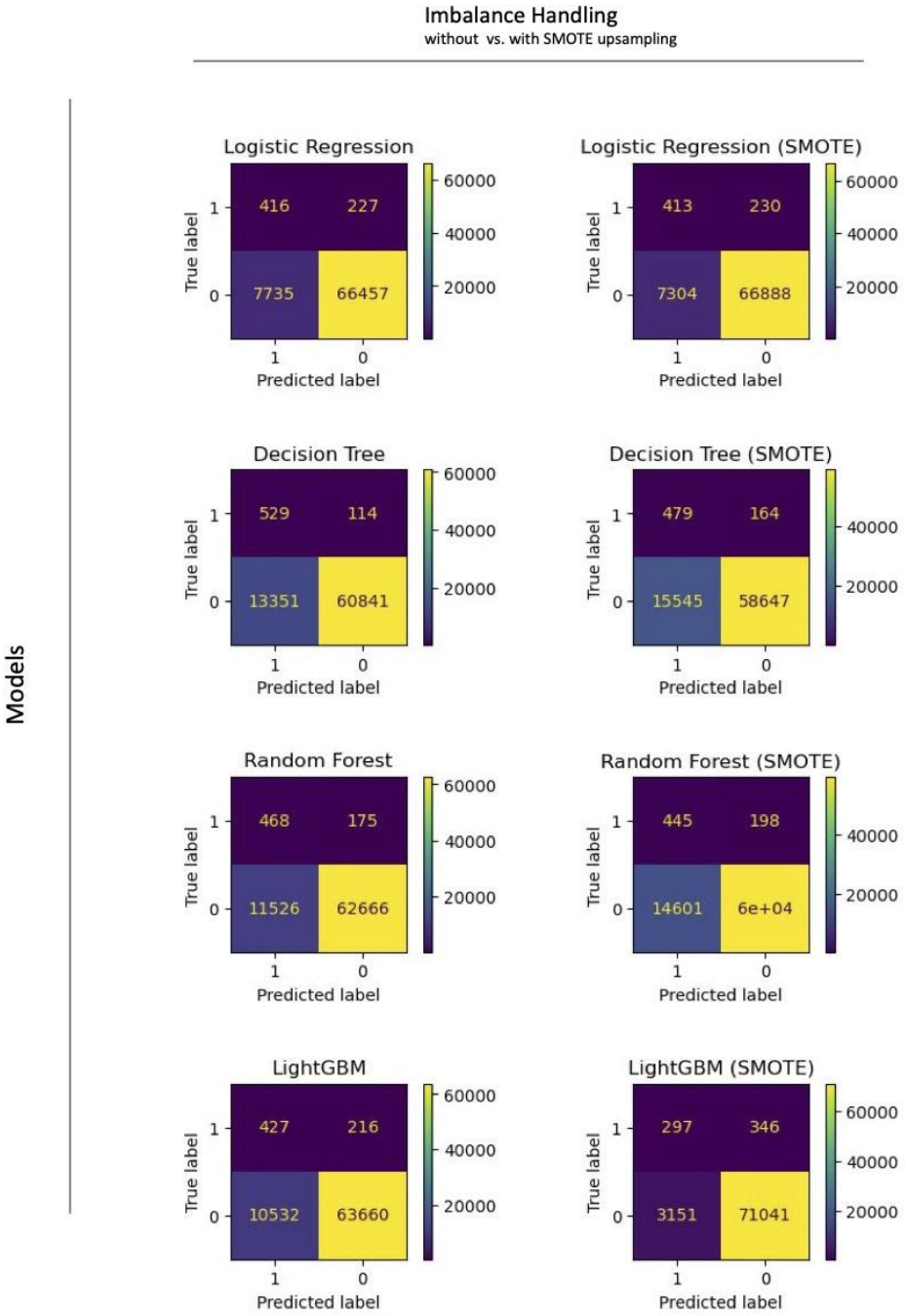
	Data Set	Metric	Dummy Majority	Logistic Regression	Logistic Regression (SMOTE)	Decision Tree	Decision Tree (SMOTE)	Random Forest	Random Forest (SMOTE)	LightBGM	LightBGM (SMOTE)
0	Export	F1	0.0	0.095	0.099	0.073	0.057	0.074	0.057	0.074	0.145
1	Export	Recall	0.0	0.647	0.642	0.823	0.745	0.728	0.692	0.664	0.462
2	Import	F1	0.0	0.267	0.082	0.140	0.153	0.267	0.234	0.319	0.740
3	Import	Recall	0.0	0.769	0.612	0.838	0.816	0.838	0.793	0.842	0.623

C.2. Export Models Feature Importance Comparison

Across 5-fold GridSearchCV Best Estimators (Seized = 1, Clear = 0)



C.3. Export Out-Of-Sample Prediction Failure Analysis Comparison
Across 5-fold GridSearchCV Best Estimators (Seized = 1, Clear = 0)



C.4. Export Feature Ablation Comparison

Across 5-fold GridSearchCV Best Estimators (Seized = 1, Clear = 0)