A Call Center Reliant on a Classifier

Fayaaz Khatri

Abstract

Reliant is an electricity provider serving Texas. It offers a product called EcoShare to customers, where carbon offsets are purchased on their behalf for a small monthly fee. Reliant would like to use a classification model to indicate if a customer is a likely candidate to enroll in EcoShare during a call center interaction. I developed a histogram-based gradient boosting classifier to model the situation, but an underlying class imbalance hinders its predictive performance.

1. Introduction

Reliant, an NRG company, is a major electricity provider in Texas. Reliant offers a variety of products alongside residential electricity service, such as protection plans, backup power and renewable products. NRG's Cross Serve Personalization Engine helps Reliant connect its customers to additional products which they may be interested in.

1.1 Background

This project focuses on one renewable product in particular - Reliant EcoShare. It offers two tiers priced at $3.95/mo and $5.95/mo, which gets tacked onto monthly electricity billing. The monthly charge goes toward purchasing carbon offsets for electricity usage, as well as funding an environmental nonprofit in Texas.

EcoShare Gold $5.95/month	**EcoShare Silver** $3.95/month
$2 EarthShare Texas donation per month	**$1** EarthShare Texas donation per month
1,000 lbs. of carbon offsets purchased per month	**500** lbs. of carbon offsets purchased per month

Existing customers can phone into Reliant's call center for questions regarding billing or service. This interaction poses an opportunity for call center agents to pitch additional products to customers. NRG's Cross Serve Personalization Engine intelligently suggests particular products for particular customers, hence equipping call center agents with their best shot at upselling an additional product during a phone interaction.

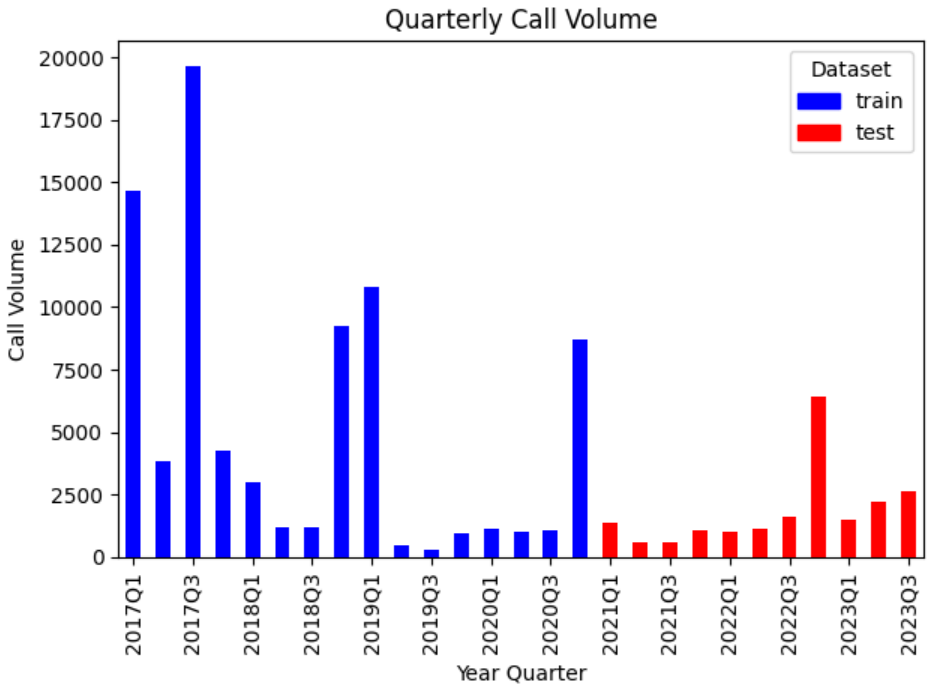
1.2 Project Objective

Reliant would like to add support for the EcoShare product in NRG's Cross Serve Personalization Engine. Reliant has provided a dataset of phone calls from existing customers that were eligible for EcoShare at the time of the call. This dataset spans from 2017 to 2020, containing over 80,000 phone call records with numerous customer attributes, along with an indicator for successful enrollment in EcoShare.

The objective is to develop a classification model that will predict the likelihood of a customer enrolling in EcoShare during a call center interaction. Reliant has also provided a test dataset of phone calls from 2020 to 2023 without the enrollment indicator. The model will be used to predict probabilities of EcoShare enrollment on the test dataset for then Reliant to validate.

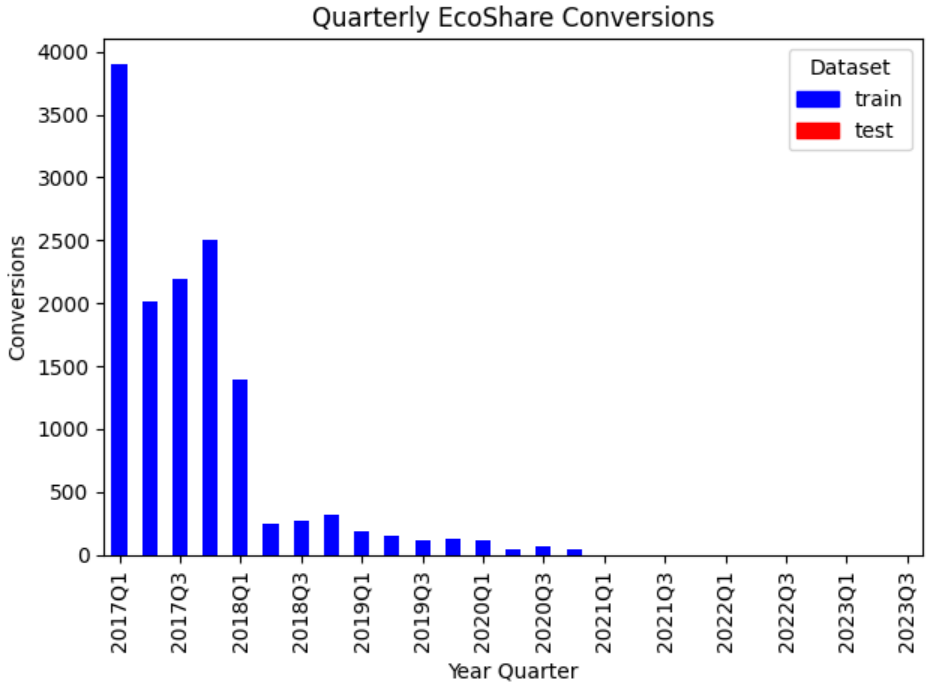
2. Data Exploration

The provided training dataset contains 30 features along with indicator variable `accept`, with a total of 83,315 call records dating from 2017 to 2020. Reliant also provided an unlabeled test dataset, containing 20,328 call records dating from late 2020 to 2023. This is approximately an 80-20 split between train (labeled) and test (unlabeled) records.



Above is a chart of quarterly call volume observed across the training and testing datasets. There are six notable peaks - five in training and one in testing.. These six peaks account for nearly 60% of call records. The peaks appear to be somewhat spread similarly with periodicity, so seasonality might be worth considering.

Below is a chart of quarterly EcoShare conversions (`accept = 1`). This is limited to the labeled training set. We immediately see that EcoShare enrollment was frontloaded in 2017 and early 2018, then conversions declined dramatically. Even with periodic spikes in call volume over the years, conversion appears to steadily decline.



There is a substantial class imbalance in the target variable in the training dataset. Only 17% of phone call records have `accept = 1`, and the chart above shows that the class imbalance gets more extreme over time. This may lead to challenges for a classification model and will have to be kept in mind when fitting and evaluating a classifier.

2.1 Data Preprocessing

Some features in the provided datasets were not immediately usable for classification modeling. The following bullet list outlines the preprocessing applied to particular fields before moving forward.

- `order_day` date object was parsed into `day_of_year` and `day_of_week` integers, which might help capture seasonality.

- `tos_flg` contained only true values, with the rest as missing, so missing values here were set to false values.
- `term_length` had two unusual values of “C&” and “MM” while the remaining were integers, so these two values were set to null.
- The following features contained categorical data, which were one-hot encoded into dummy variables, including a level for missing values.
    - `dwelling_type_cd`
    - `product_type_cd`
    - `tdsp`
    - `segment`
    - `pool`
    - `risk_level`
    - `load_profile`
- The following features were especially sparse, so these were dropped from the datasets.
    - `deposit_onhand_amt` - missing in 93.9% of records
    - `home_value` - missing in 74.6% of records

2.2 Feature Engineering

Some features are categorical variables with very high cardinality, for example - city names and electric plan names. This can lead to poor runtime and unreliable results in classifiers. Furthermore, some key business context is not explicitly captured in the provided dataset. This is a call center with humans interacting on both sides, so some situational and behavioral features can be deduced, and better yet they could be meaningful in modeling. Below is a summary of feature engineering performed to address these topics.

- `zipcode` was geocoded into `latitude` and `longitude` features. Though appearing numeric, zip codes have no meaningful numeric interpretation. Mapping these to latitude and longitude brings visibility to relative position and proximity between customers.
- `sap_productname` describes the name of the electricity plan. This contains over 500 unique entries in the training dataset. To reduce the high cardinality, I looked at term frequency across all values in this field. I took the top 15 frequently occuring meaningful words and created 15 dummy variables to flag an entry containing a word, enumerated below.
    - `easy`
    - `flex`
    - `secure`
    - `weekends`
    - `conservation`
    - `save`
    - `simple`

    - `solarsparc`
    - `pollution free`
    - `wind`
    - `business`
    - `apartment`
    - `smart`
    - `discount`
    - `solar`
- Some historical and behavioral customer metadata can be deduced when a customer has multiple records in the dataset over time.
    - `customer_id` and `meter_id` can be used to determine the number of meters tied to a customer (`customer_num_meters`).
    - A prior record with `accept = 1` for a customer means that the customer was previously enrolled for EcoShare, but may have since canceled it (`prev_enrolled`).
    - Prior records for a customer means that call agents may have previously attempted to upsell EcoShare (`prev_attempts`).
- Daily call volume can be a huge indicator about customer behavior. Perhaps an outage or weather event has many customers calling in for support. The high volumes could impact opportunities for call agents to mention EcoShare (`total_calls_on_day`).
- Enrolling in EcoShare ultimately increases a customer's bill. Some insight into a customer's income could be a meaningful factor. I used a 2020 median household income by Texas county dataset[1] and combined it with the provided dataset based on service address `county` (`county_median_household_income`).
- A `load_profile` ending in `PV` suggests that a customer has solar panels and a bidirectional meter. There could be a relationship between interest in solar power and interest in EcoShare, so I added an indicator variable for this (`load_profile_pv`).

After data preprocessing and feature engineering are applied, the dataset contains 69 features.

3. Developing a Classifier

This section will focus on the provided training dataset, since class labels (`accept = {0, 1}`) will be needed to train a model. I split the labeled dataset into a training set and validation set using an 80-20 ratio, aligning with sizes of the full training and testing sets. The business context here is time-series natured, so the split was not randomized. The first 65,052 records of the provided training dataset were used to train, and the last 16,263 records were used to validate.

[1] https://www.texascounties.net/statistics/householdincome2020.htm

3.1 Model Selection

I limited the model selection search to the five candidate classifiers and their implementations in `scikit-learn`, listed below:

- Random Forest Classifier
- Histogram-based Grading Boosting Classifier
- XGBoost Classifier
- Logistic Regression
- K-Nearest Neighbors Classifier

The first three are decision tree ensembles, which can be effective in capturing complex structures while avoiding overfitting. I figured a linear model like logistic regression is worth considering too, in case there's an underlying linear relationship between features and EcoShare acceptance. Lastly I considered a proximity-based classifier with K-nearest neighbors, since it could be reasonable for like customers to behave similarly toward additional products like EcoShare.

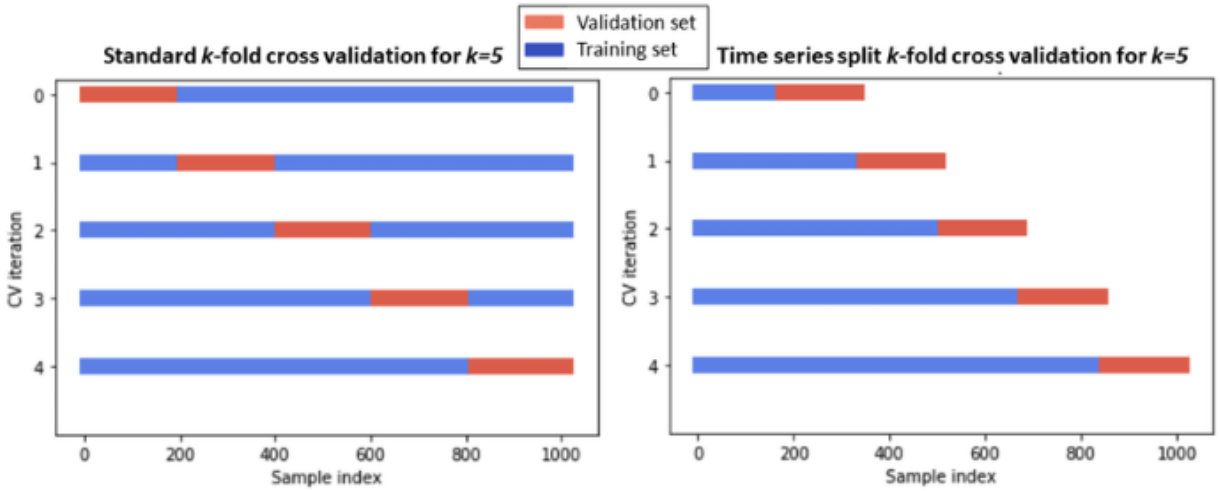
Since some of the models listed above do not support missing values, some data imputation was required for model selection.

- `term_length` - this feature had two bad/unexpected non-numeric values that were set to null.
- `curr_usage` - this feature had about 10% missing values

To impute these missing values, I trained a K-nearest neighbors imputer on the training set. This imputer was applied to the validation set as needed. If applicable for a classifier, I set the parameter `class_weight='balanced'` to account for the class imbalance in the training dataset target variable.

To train and hyperparameter tune the candidate classifiers, the training set was given a 5-fold time-series split for cross validation. Unlike the usual randomized scheme for cross validation, the time-series split maintains the chronological order of records. Folds are also cumulative, in that iteration `n` using the first `n` splits to train and validate. This scheme avoids data leakage of future records - meaning that a model was trained with future information to infer the past. The time-series split is illustrated below, in a sample dataset of 1,000 records.[2]

[2] Predicting the Price of Crude Oil and its Fluctuations Using Computational Econometrics: Deep Learning, LSTM, and Convolutional Neural Networks - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Classical-k-fold-cross-validation-vs-time-series-split-cross-validation_fig1_355889701 [accessed 21 Nov, 2023]



For each candidate classifier, I defined a range of values to consider for model hyperparameters. I performed a random search time-series based cross validation with 25 iterations, meaning that 25 different combinations of hyperparameter values were considered for each classifier. For more details on the hyperparameters considered, please refer to the script `model_selection.py`. To score the iterations of cross validation, I considered the following three metrics.

- Balanced accuracy - arithmetic mean of recall across classes
- F1-score - harmonic mean of precision and recall
- Average precision - mean of precision, weighted by increase in recall by incremental threshold

The three metrics listed above strike a good balance between precision and recall. Precision and recall tied to class label 1 ought to drive the model selection decision, since they will not be skewed by the underlying class imbalance in the target variable. Given that 83% of phone calls upsells do not lead to EcoShare enrollment, a trivial classifier could infer all records as class 0 and it would be expected to achieve about 83% accuracy, which sounds great, but it is clearly misleading!

Below is a table summarizing the best average score achieved by each candidate classifier across all three scoring metrics during time-series based cross validation.

Time-Series Cross Validation Results			
Model	Balanced Accuracy	F1-score	Average Precision
Random Forest	0.705	0.432	0.471
Histogram-based Gradient Boosting	0.710	0.482	0.486
K-Nearest Neighbors	0.626	0.384	0.317
Logistic Regression	0.684	0.414	0.305
XGBoost	0.648	0.378	0.46

Histogram-based gradient boosting achieved the top average score across all three metrics. It just slightly edged random forest, but outperformed the other models with some margin. It also trains exceptionally quickly compared to the other models because of its histogram approach. Continuous features are slotted into at most 256 bins, much like a histogram. This vastly simplifies the splitting decision of a tree node, since it has at most 256 values to consider instead of a range of continuous values. With exceptionally quick train time and the best performance across all three metrics, I decided to select the histogram-based gradient boosting classifier as my final model.

3.2 Model Evaluation

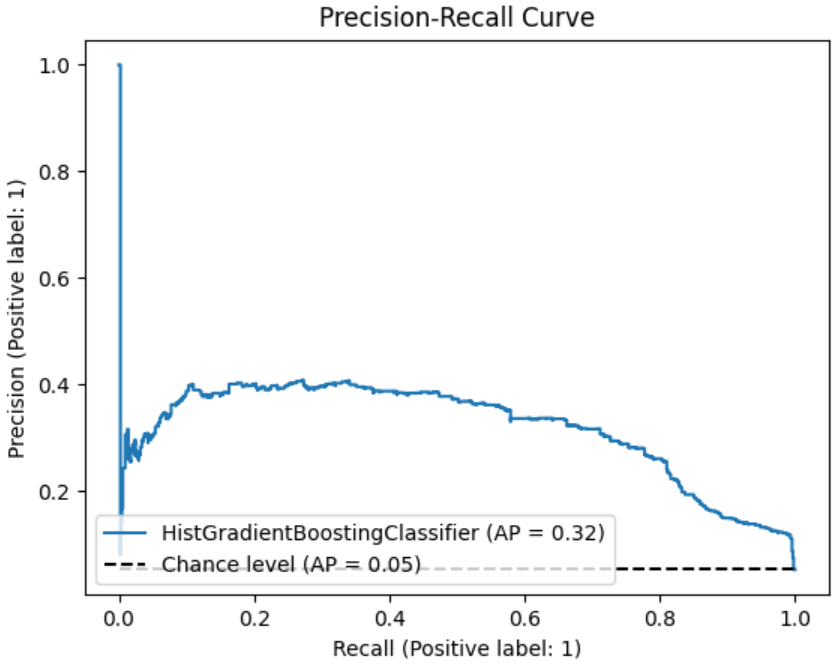
To evaluate the histogram-based gradient boosting classifier, I fitted the model using the split training set - the first 65,052 records of the provided labeled dataset. This classifier has support for missing values, hence no need to impute missing values this time. The model then inferred class labels on the remaining 16,263 records, using 50% as a threshold. The following table summarizes classification metrics for both class labels.

Classification Metrics on Validation Set				
Class Label	Precision	Recall	F1-score	Support
0	0.96	0.98	0.97	15,410
1	0.40	0.28	0.33	853

The model performs exceptionally well to predict class label 0, corresponding to a customer not enrolling in EcoShare (`accept = 0`). The results get shakier for the prediction of primary interest - class label 1 (`accept = 1`). The support, meaning the spread of true labels across classes, is more imbalanced in the validation set than the training set. Only about 5% of phone calls led to EcoShare enrollment. This is a potentially worthwhile difference to evaluate against, since the rate of EcoShare acceptance has shown to decline over time.

The class imbalance drives the stark difference in performance seen between the two classes. It is important to focus on classification metrics that do not get clouded by the amazing performance of class label 0. For example, the AUC of the ROC curve is 0.91, which seems spectacular because the poor performance of class label 1 is not fairly balanced against the massive class label 0.

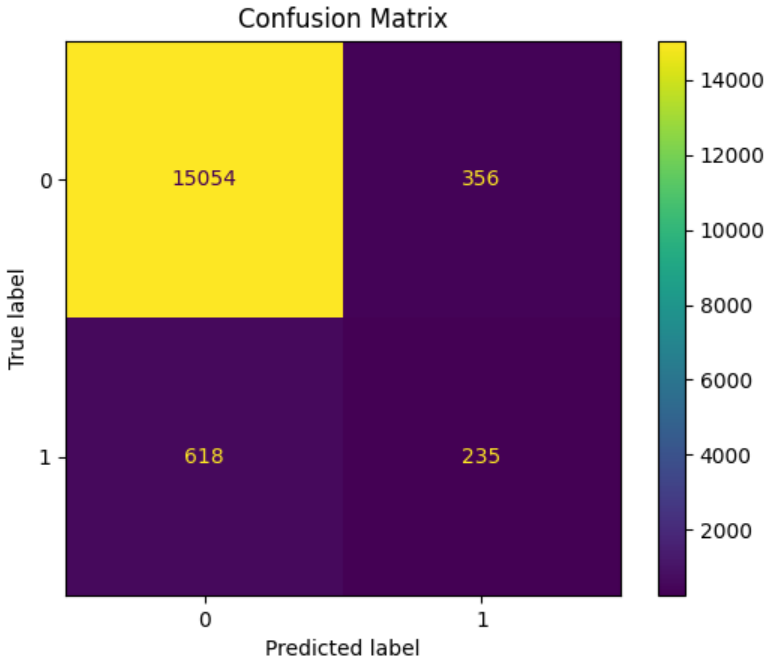
The precision, recall and F-1 score for class label 1 capture the underlying business goal here - predicting successful enrollment of EcoShare. A worthwhile alternative to the ROC curve is the Precision-Recall curve because it is limited to the positive class label 1. Class imbalance is no longer relevant, as it won't be influenced by class label 0 at all. It is plotted below at varying classification thresholds.



It is interesting to see that precision peaks at about 0.4, and recall can range from 0.1 to 0.7 without sacrificing precision. A higher recall means fewer false negatives. False negatives are costly in the business context, suggesting that a customer who is likely to enroll in EcoShare was not given a sales pitch from a call agent. Recall may be worthwhile to increase, and fortunately precision doesn't have to be compromised much.

Average precision (AP) can be considered analogous to the AUC for the Precision-Recall curve. AP is the mean of precisions at each threshold, weighted by the increase in recall from the prior threshold. The AP of the classifier on the validation set is 0.32. This is certainly better than a chance-level random classifier with an AP of 0.05, but there is still room for improvement. The classifier offers some predictive insight, but unfortunately with more mistakes than correct

predictions on class label 1. This is shown below in the confusion matrix - highly specific but not sensitive.

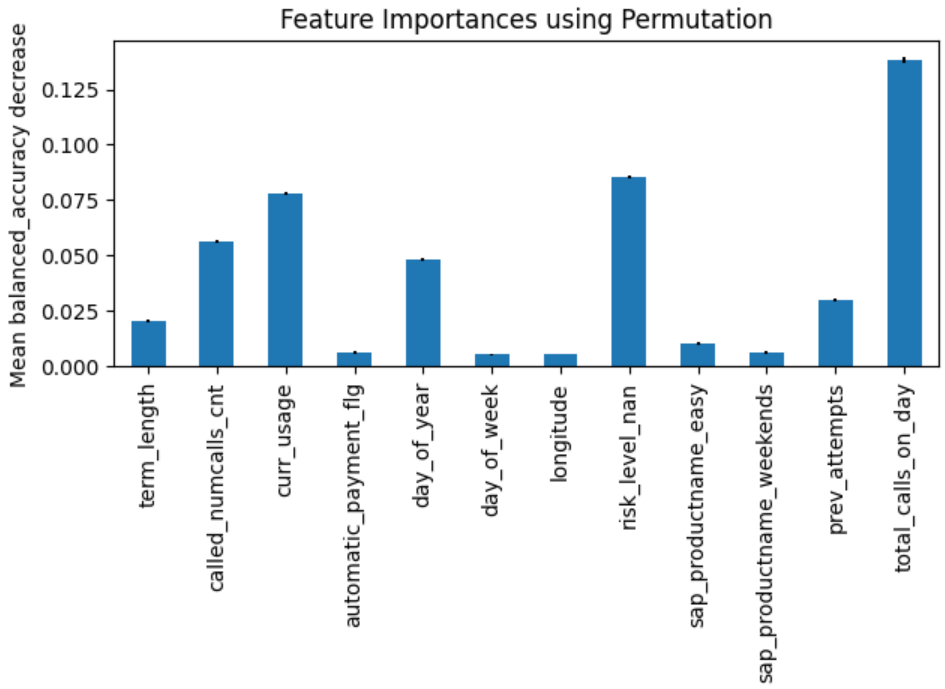


3.2 Feature importance

The histogram-based gradient boosting classifier is somewhat of a black box model. As a non-parametric model, there's no coefficients directly assigned to features during fitting. It is valuable to learn about features and their importance in a model, as it can reveal business insight and underlying relationships.

I used a permutation approach to estimate feature importance from the classifier. For a given feature, its values are randomly permuted across all records. The permuted dataset is then tested against the classifier. The change in scoring metric is noted, and this is repeated five times across all features individually. The average decrease in score gives us some insight into the importance of a feature.

I chose to use balanced accuracy as the scoring metric here, which is the arithmetic mean of recall for the class labels. Recall is worthwhile to optimize because a false negative can be a costly missed opportunity for EcoShare enrollment. Below is a chart of the top mean decreases in balanced accuracy across all the features from permutation.



Of the 69 total features, the 12 shown above had the highest decline in balanced accuracy. The biggest impact comes from `total_calls_on_day` . It is interesting to see that metadata about the call center itself is the most important feature. I figure that a high call volume day may lead to low acceptance of EcoShare because customers may be preoccupied with service outages. There's a good chance a call agent has no time to mention EcoShare as well. Next comes `risk_level_nan`, which indicates a missing risk level for a customer. This could imply that a customer is relatively new to Reliant, and hence may be more open to additional products. After that is `curr_usage`, the amount of electricity consumed during the previous billing cycle. Feature importance starts to become less impactful at this point, but some notable ones are `called_numcalls_cnt` (# of times a customer called the call center in the past month), `day_of_year` (suggesting a possible seasonality, possible relationship to weather), and `prev_attempts`. It's interesting to see that just 6 of 69 features truly drive the classifier's predictive insights.

4. Closing Remarks

I retrained the histogram-based gradient boosting classifier on the entire provided training data set. I used the classifier to infer predicted probabilities on the provided unlabeled test set and delivered predictions to Reliant for validation.

From my own validation, this classifier does a great job in telling us where to not place efforts. It can determine customer calls that will not result in EcoShare enrollment with spectacular performance. Reliant is more interested in the opposite, and the classifier is somewhat mediocre on that front. There is certainly some insight behind it, as it greatly outperforms a trivial random classifier, but it does leave something to be desired.

I hoped engineered features like `median_household_income` would show a strong relationship, but unfortunately that did not materialize. I also had a good feeling about customer metadata features (`prev_enrolled`, `prev_attempts`), but these showed little importance. One would think that an `ev_driver` could be interested in EcoShare, but again that is not quite what we saw.

Reliant's true business motivation is to increase enrollment in EcoShare. The particular context of a call center might be a limiting factor in this goal. The ever increasing class imbalance over time suggests that EcoShare enrollment from a call center interaction is a rare event. Either customer preferences have shifted away from renewable products like EcoShare, or the call center is a less than ideal opportunity to upsell it. My bet is on the latter. The most important feature for the classifier is `total_calls_on_day`, which is call center metadata and not customer related at all. The classifier accounts for the class imbalance and performs the best it can, but I believe there's a disconnect between the business goal and the call center. Ultimately I suggest Reliant to reevaluate the call center as an avenue to sell additional products.