Recidivism Forecasting Challenge

Anuar Assamidanov

Problem Statement

The Recidivism Challenge aims to improve the ability to forecast recidivism using person- and place-based variables with the goal of improving outcomes for those serving a community supervision sentence. We hope through the Challenge to encourage discussion on the topics of reentry, bias/fairness, measurement, and algorithm advancement. In addition to the Challenge data provided, NIJ encourages contestants to consider a wide range of potential supplemental data sources that are available to community corrections agencies to enhance risk determinations, including the incorporation of dynamic place-based factors along with the common static and dynamic risk factors. NIJ is interested in models that accurately identify risk for all individuals on community supervision. In order to do this, contestants will need to present risk models that recognize gender specific differences and do not exacerbate racial bias that may exist.

Under this Challenge, NIJ is providing a large sample accompanied with rich data amendable for additional data to be paired with it. NIJ expects that new and more nuanced information will be gained from the Challenge and help address high recidivism among persons under community supervision. Findings could directly impact the types of factors considered when evaluating risk of recidivism and highlight the need to support people in specific areas related to reincarceration. Additionally, the Challenge could provide guidance on gender specific considerations and strategies to account for racial bias during risk assessment.

The Challenge uses data from the State of Georgia about persons released from prison to parole supervision for the period January 1, 2013 through December 31, 2015. Contestants will submit forecasts (percent likelihoods) of whether individuals in the dataset recidivated within one year, two years, or three years after release.

Project Overview

In this project, I analyzed and predicted the likelihood of recidivism using profile data. To accomplish that, I performed Logistic regression, Random Forest Classifier, XGBoost, LightGBM, and Catboost algorithms and evaluated performance. I divided the project into six main parts. I went through the Exploratory Data Analysis, Feature Engineering, Model Building, Model Evaluation, Feature Importance, and Inference.

Exploratory Data Analysis

The dataset has 53 columns and 18028 rows. The data includes individual- and place-based variables that capture the supervision case information, prison case information, prior Georgia criminal history, prior Georgia community supervision history, Georgia board of pardons and paroles conditions of supervision, and supervision activities. The columns in the data are deemed to be a proxy for the already-established profile of respondents. They can be used to accentuate the salience of person-based and place-based recidivism forecasting.

Our primary outcome variable is recidivism. The table below shows the percentage and number of recidivated people in the sample of the 18028 population. It occurs the percent decreases by each year from 29.8 to 19 percent. Overall, within three years, 57.9% are recidivated.

	Recidivism Year 1	Recidivism Year 2	Recidivism Year 3	Overall
Number of Recidivated People/Percentage	5377 (29.8%)	3253 (25.7%)	1791 (19%)	10421 (57.9%)
Male	4920 (31.1%)	2889 (26.5%)	1601 (20%)	9410 (59.5%)
Female	457 (20.6%)	364 (20.7%)	190 (13.6%)	1011 (45.6%)
Black	3198 (31%)	1830 (25.7%)	1048 (19.8%)	6076 (58.9%)
White	2179 (28.2%)	1423 (25.7%)	743 (18.1%)	4345 (56.3%)
Total Number of Observation	18028	12651	9398	18028

Table 1. Descriptive Statistics of the data

I divide the table into gender and race categories to see how the recidivism varies on those variables of interest. It occurs that the male population is substantially higher than the female. Recidivism rate conditioning on gender differs 11-7% each year, meaning that the male population tends to re-offend more than the female population. Furthermore, the number of black populations recidivated is higher than the white population. However, the percentage of recidivism with race category has almost the same value in each year. To sum up, we can see that recidivism within three years exceeds 40% for each variable in Table 1.

Feature Engineering

In this part, I will explain how I extracted features from raw data, potentially improving the performance of machine learning algorithms. Overall, in this project, I did not find that feature engineering produces substantial improvement in the results, meaning that the power of model learning and extracting insight was better than the human manipulation of the data. I have done all feature engineering to deal with missing data, converting booleans into integers, converting categorical variables into a dummy, and converting decimals into integers.

Initially, I converted the outcome variables into a dummy variable. It will transform our research question into the likelihood of recidivism based on the given variables. This dummy transformation gives us a great opportunity to effectively apply classification algorithms like Logistic Regression, Random Forest, Gradient Boosting, Neural Networks, etc.

Regarding missing values, to my knowledge, lightGBM and CatBoost will ignore missing values during a split, then allocate them to whichever side reduces the loss the most. However, for other models, manually dealing with missing values improved model performance. It seems like if I set missing values to -99, it produces the best results. I did not see any improvement in imputing missing values to mean, median, or model.

Model

In this part I will briefly explain the machine learning model that I used in this project.

Logistic Regression

Logistic Regression is a transformation of a linear regression using the sigmoid function. The vertical axis stands for the probability for a given classification and the horizontal axis is the value of x. It assumes that the distribution of y|x is Bernoulli distribution. (1)

Random Forest

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result. (2)

Xgboost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. (3)

LightGBM

LightGBM is a gradient boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient with the following advantages: Faster training speed and higher efficiency, Lower memory usage, and Better accuracy, Support of parallel, distributed, and GPU learning, and Capable of handling large-scale data. (4)

Catboost

CatBoost is an algorithm for gradient boosting on decision trees. It is developed by Yandex researchers and engineers, and is used for search, recommendation systems, personal assistant, self-driving cars, weather prediction and many other tasks at Yandex and in other companies, including CERN, Cloudflare, Careem taxi. It is in open-source and can be used by anyone. (5)

Model Building

In this section, I will use the prediction of the third year as a case study to explain how I approached this project. So, the outcome variable is the recidivism of the respondent after three years post-incarceration. I built several machine learning models and analyzed their results.

I start with the most basic, a logistic regression, which predicts the likelihood of recidivism. Logistic regression would serve as our baseline. Following that, I have performed Random Forest Classifier, Xgboost, LightGBM, and Catboost algorithms. To evaluate training set performance, I have implemented Stratified K-fold Cross-Validation Method. This technique is a variation of KFold that returns stratified folds. Since there are many categorical variables in our data, the folds preserve the percentage of samples for each categorical variable.

To boost the performance of the algorithms, I have implemented a hyperparameter tuning exercise. I applied the grid search method, which is a process that searches exhaustively through a manually specified subset of the hyperparameter space of the given algorithm. I used the "pruning" technique to stop training earlier when the learning curve was much worse than the best-known result. The algorithm selected parameters based on the ROC-AUC evaluation metrics. I gathered models with optimized hyperparameters into an array. Overall, I have made 19 models: one Logistic regression, three Random Forest Classifiers, and five LightGbm, Xgboost, and Catboost.

I incorporated these models to make one generalized prediction called the stacking method. Stacking is a way of combining multiple models that introduces the concept of a meta learner (6). This classifier fits base classifiers, each on random subsets of the original dataset, and then aggregates their predictions to form a final prediction (either by voting or by averaging). The point of stacking is to explore a space of different models for the same problem. The idea is that you can attack a learning problem with varying types of models capable of learning some part of the problem but not the whole space. So you can build multiple different learners and use them to make an intermediate prediction, one prediction for each learned model. Then you add a new model which learns from the intermediate predictions the same target. This final model is said to be stacked on top of the others, hence the name. Thus, you might improve your overall performance, and often you end up with a model which is better than any individual intermediate model (7). The main takeaway from all these steps is to create a generalized outcome that will not overfit your training set.

I split the original dataset into a Training and Holdout dataset. Let training go onwards into the upcoming loop and save holdout until the last part in the forthcoming loop. I made a for loop with KFold Cross-Validation where k=5. In each iteration, I split the Training dataset into training and validation datasets. I called them X_train, y_train, X_valid, and y_valid. The red parts in Figure 1 represent X_train and y_train, while the green represents X_valid and y_valid. I trained the current model on X_train and y_train.

I made predictions on the test dataset X_valid and evaluate it with the y_valid. I extended an array of the predictions for the whole training dataset. I called them out of sample predictions. I run these out of sample predictions for all algorithms stated above. Eventually, I got 19 out of the sample predicted arrays. I used them as new features for the new training dataset composed of 19 features and outcome variables. Then, I run Ridge Regression to predicted based on these input variables. I also did the Kfold cross-validation technique the same as above, but I run a fitted model on the Holdout dataset. I have chosen five folds and created five predicted outcome variables from the holdout dataset. After that, I took a mean of these five predicted outcome variables, my final predicted recidivism probability.

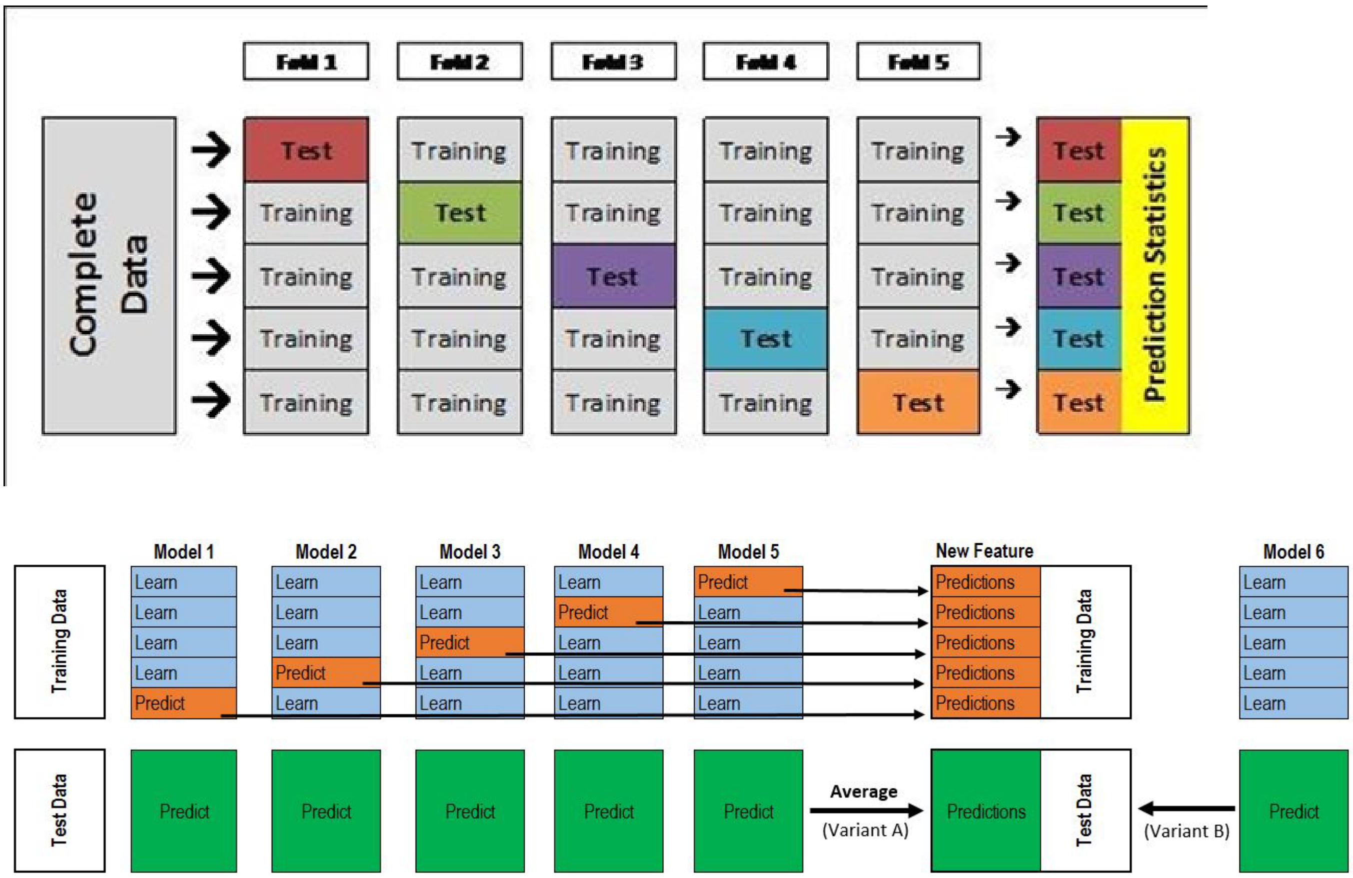

Figure 1. The illustration of K-Fold Cross Validation and Stacking method

Model Evaluation

To achieve a decent score and find out the best hyperparameters, I run an overall 16015 trials using the cutting-edge Python Library of Optuna. Then, I have chosen the best one from Logistic Regression, three from Random Forest, and five from LightGBM, CatBoost, and

XGboost. The decision from the chosen numbers comes from personal experience working with these algorithms.

In this part, I would like to explain the results of each model and what score I got. From Table 2, we can see that the best performance comes from CatBoost. From my observation, I noticed that fairness and accuracy results are highly correlated with the AUC score. Therefore, we can rely on the AUC score, which comes with each algorithm in Python Library.

	Model I	Model II	Model III	Model IV	Model V	Trials
Logistic Regression	0.668					15
Random Forest	**0.681**	0.6801	0.6804			100
LightGBM	0.681922	0.681967	0.682350	0.682453	**0.682980**	500
CatBoost	**0.708976**	0.708665	0.703801	0.701938	0.703322	500
XGBoost	**0.6889**	0.6881	0.6879	0.6875	0.6871	500

Table 2. AUC result of the algorithms

As I previously mentioned, I choose all of these algorithms out of sample results to run the Stacking method. From this method, we can see that overall it is lower than the best performing Catboost method. Assuming stacking will take the best part of each algorithm and performs generalized result, I have chosen to stack as my outcome.

	Fold I	Fold II	Fold III	Fold IV	Fold V	Overall Metrics
Stacking	0.6843	0.6956	0.7004	0.7126	0.70731	0.6996

Table 3. AUC score of Stacking Method

Inference

	Feature I	Feature II	Feature III	Feature IV	Feature V
Logistic Regression	Age at Release 18-22	Age at Release 48 or older	Gang Affiliated	Age at Release 43-47	Age at Release 23-27
Random Forest	Jobs Per Year	Percent Days Employed	Avg Days per Drug Test	Prior Arrest Episodes PP Violation Charges	Supervision Risk Score First
LightGBM	Avg Days per Drug Test	Jobs Per Year	Percent Days Employed	Supervision Risk Score First	Residence PUMA
CatBoost	Jobs Per Year	Avg Days per Drug Test	Percent Days Employed	Age at Release	Supervision Risk Score First
XGBoost	Prior Arrest pp violation Charges	Gang Affiliated	Prior Conviction Episodes Misd	Violation Instruction	Prior Arrest Episodes Misd

Table 4. AUC result of the algorithms

In this part, I would like to go over the feature importance analysis. According to the table, we can see that each model has its own top five most important features. Feature importance is determined by a mean decrease in impurity for decision tree-based models and logs odd coefficients for Logistic Regression. As we can see, the LightGBM and CatBoost have more similarities in features, but the order is not the same. Since the core base of these models is gradient boosting techniques, thus AUC scores are also close.

Furthermore, Logistic Regression has completely different features, which is reasonable since it has low predictability than the gradient boosting techniques. As can be seen from Table 4, job-related variables like the number of jobs per year and percent days employed have the highest correlation. The idea comes together with Kevin Schnepel (2018) where he examines "good jobs" and asks how their presence influences recidivism. Prior criminal history records such as prior misdemeanor and violation charges have shown high importance in the XGboost model. Average Days on Parole Between Drug Tests tend to be helpful to achieve good predictability in Catboost and LightGBM. Overall, the main take from these models is these features are highly correlated with the outcome variable of recidivism.

Reference

1. Sigmoid Function - an overview | ScienceDirect Topics. https://www.sciencedirect.com/topics/computer-science/sigmoid-function
2. Random Forest Algorithms: A Complete Guide | Built In. https://builtin.com/data-science/random-forest-algorithm
3. XGBoost Documentation — xgboost 1.5.0-dev documentation. https://xgboost.readthedocs.io/
4. Welcome to LightGBM's documentation! — LightGBM 3.2.1.99 .... https://lightgbm.readthedocs.io/en/latest/index.html
5. CatBoost - open-source gradient boosting library. http://catboost.ai/
6. Ensemble Methods. Essential Machine Learning Concepts to Know. https://medium.com/analytics-vidhya/ensemble-methods-b644f9c94bc1
7. Stacking in Machine Learning - GeeksforGeeks. https://www.geeksforgeeks.org/stacking-in-machine-learning/
8. Good Jobs and Recidivism - Kevin Schnepel. https://kschnepel.github.io/research/2018-GoodJobsRecidivism-paper-6