Prediction algorithm for crime recidivism

Julia Andre, Luis Ceferino and Thomas Trinelle
Machine Learning Project - CS229 - Stanford University

Abstract

This work presents several predictive models for crime recidivism using supervised machine learning techniques. Our initiative was focused on providing insights which would help judges to make more informed decisions based on the analysis of individuals' proneness to recidivism. Different approaches were tried and their generalized error were computed and compared using cross validation methods. Models are trained on two large data set collected from the Inter-university Consortium for Political and Social Research (ICPSR).

Introduction

Today, the United States have one of the highest recidivism rate in the world: with 2.3 billion people in jail, almost 70% of the prisoner will be re-arrested after their release. This poses a serious problem of safety, and proves that we don't make the decision that really make us safer. Judges, even though they have good intentions, make decision subjectively. Studies show that high-risk individuals are being released 50% of the time while low risk individual are being released less often than they should be (Milgram (2014)). Ideal would be to detain an offender for precisely the right amount of time so that he is not re-arrested after his release, but in the mean time does not spend excessive time in prison. With machine learning tools we can produce accurate predictive models based on various factors such as age, gender, ethnicity, employment. Detecting patterns in recidivism would provide supporting arguments for judges to determine the appropriate sentence, which will decrease safety risks while trying to avoid over-punishment.

The purpose of this project to use data and analytics to transform the way we do criminal justice. Using supervised learning we can design a predictive model for recidivism trained on historical data collected in the US. This decision making tool will help the judges determine whether a new offender is dangerous or not, by giving him a "recidivism score".

Problem formulation

This study provides element of answers to the following questions:

(1) Can we create an accurate predictive model to detect individual likely to commit recidivism? If yes, what would be its accuracy?
(2) If a judge, was to have a very limited access to data, what would be the most important features he would want to collect on an individual to make a reliable judgement?

Model development

Our analysis consisted of the following steps :

(a) **Data Acquisition**: Crime and felonies are sensitive information which added latency for our team to collect. The data required for machine learning applications needed to comply with two main characteristics: (1) to be large enough such that the machine learning techniques can converge to stable parameters; (2) contain relevant features to the problem we want to evaluate. Bearing in mind these requirements, our team searched online information from different penitentiary institutions and research centers in the US. Additionally, our team contacted Himabindu Lakkaraju, a PhD student in the CS Department who is working on artificial intelligence applied to human behaviours related to criminology for feedback. After extensive exploration, our team found a relevant database in the Inter-University Consortium for Political and Social Research (ICPSR) Website. This data set was collected by Smith and Witte (1984), and has information of two cohorts of inmates that were released in 1978 and 1980 from the prison of North Carolina. Note that publicly available data-sets are ancient, due to prescriptions, which means they are often numerical re-transcription of manually stored data.

(b) **Data pre-processing**: The format of the data required pre-processing to transcript them from there original format (SAS or SPSS) to more simple .csv file format.

(c) **Feature extraction**: The most important feature collected for both cohorts was whether or not individuals committed recidivism after release. In total, there were 19 features per individual: race, alcoholism problems, drug use, after-relase supervision (i.e. parole), marital condition, gender, conviction reason (i.e. crime or felony), participation in work release programs, whether or not the conviction was against property, whether or not the conviction was against other individuals, prior convictions if any, number of years of school, age, time of incarceration, time between the release day and the record search, whether they committed recidivism in the previous time span, the time span from the release month to the recidivism date, and a flag indicating if the individual's file was part of the training set of the study in 1984.

(d) **Preliminary analysis**: We have run direct analyses using existent libraries in Python (numpy, scipy and scikit-learn) and Matlab (liblinear and libsvm). Diagnostics were run taking into account the first 16 features of the data-set (excluding time to first recidive and file category), the output was whether or not a released convict would commit recidivism.

Analysis and results

Our initial results running simple diagnostics on both Matlab libraries and Python libraries show an excellent agreement. Preliminary diagnostics using simple Logistic Regression and linear SVM showed a testing error rate hardly below 36% which remains fairly high given the size of our data-set (about 12500 training points, for about 5500 testing points).

In light of the previous observations we decided to explore the following different next steps:
- Run a Principal Component Analysis to study the contribution of each features to the principal vectors.
- Explore different algorithms, some outside the ones covered in class, to identify the most performing ones.
- Draw learning curves using different algorithm to provide insight on potential error mitigation strategies.
- Study the feature distribution among the data-set as well as engineer features to understand importance and correlations.

Principal Component Analysis

Our team acknowledged the need for understanding how the features relate to each other. Moreover, our team realized in the preliminary analysis that the SVM algorithm was very expensive in terms of computational time. Consequently, we team explored a Principal Component analysis in the data set using the complete set of features to reduce the problem dimensionality, and to understand which set of features carried the largest variance of the problem. The features were normalized to have mean 0 and standard deviation 1. Figure 1 shows that the feature 'Age' dominates the first component, whereas the feature 'Time Served' dominates the second one. Also, results show that the first component explains nearly 95% of the total variance, whereas the second component explains 4% of the total variance.

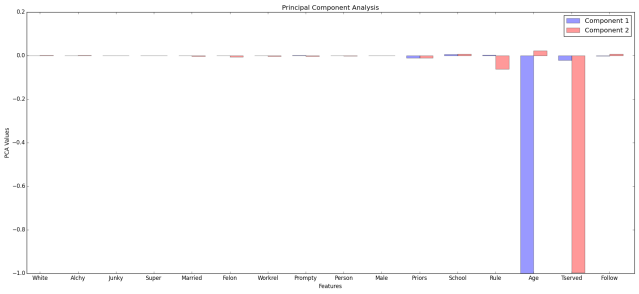

Figure 1: Contributions of Features on PCA. Blue: 1st Comp. Red: 2nd Comp.

Furthermore, our team did a transformation of the feature space into the first component subspace. A preliminary analysis with linear-kernel SVM revealed that the test error was 37% using a 5-fold cross validation. Similarly, using the first two components, the test error was 40%, and for the first three components, the test error was 42%. These results indicated that SVMs did not perform better than the simple Logistic Regression, and that the computational time involved in SVMs was hundreds of times larger than Logistic Regression. We consensually decided to stop using SVMs with different kernels and focus on exploring different algorithms.

Algorithm Exploration

Direct Runs

Our team used a set of Machine Learning algorithms to verify which would perform the best in terms of accuracy of the prediction. Our team chose the algorithms based on the material covered in the class CS229 as well as common ones such as Random Forest Gradient Boosting recommended by Lakkaraju. These algorithms are usually good predictors in cases on which the data set has several classification features. Table 1 shows the algorithms that were used in this part of the project, and the associated test and training errors in a 5-fold cross-validation. These results were calculated using the default parameters of the algorithms in the sklearn library of Python, i.e. the Random Forest and the Gradient Boosting Algorithm were run using 10 trees and until there was only one element on each leaf. Perceptron and Logistic Regression algorithms did not have a relevant parameter user-definition.

Algorithm	Training Error	Test Error
Perceptron	0.389	0.390
Logistic Regression	0.352	0.356
Random Forest	0.02	0.36
Gradient Boosting	0.30	0.32

Table 1: Algorithms in the Direct Run

The results in Table 1 indicate that Gradient Boosting is the algorithm with the least Test error. Nevertheless, to be conclusive about the supremacy of Gradient Boosting over the other algorithms, our team decided to evaluate the sensitivity of the results to the parameter-definition of the algorithms.

Parameter Estimations

The previous table proved the efficiency of algorithms using trees. We therefore pursued that effort tried to find the optimal parameter settings for our problem. Figure 2 shows how the number of estimators (i.e. number of trees) affects the test error in the Random Forest and Gradient Boosting algorithms. It can be observed that with a greater number of estimator, the error decreases. Yet, there is a threshold because using an inconsiderate number of estimators increases significantly the computational time. We therefore decided to use 40 estimators as a good balance between a reasonable test error and running time.

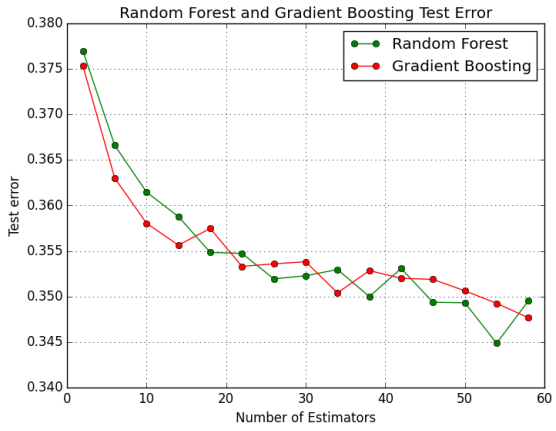

Figure 2: Sensitivity of Test Error with respect to Number of Estimators

Considering 40 estimators for both algorithms, we then plotted the variation of the test error for the maximum depth of the trees (i.e. of sub-divisions). We also set the number of elements per leaf at 20 elements as a limit to subdividing the selected sub-set of data. Figure 3 points out that the

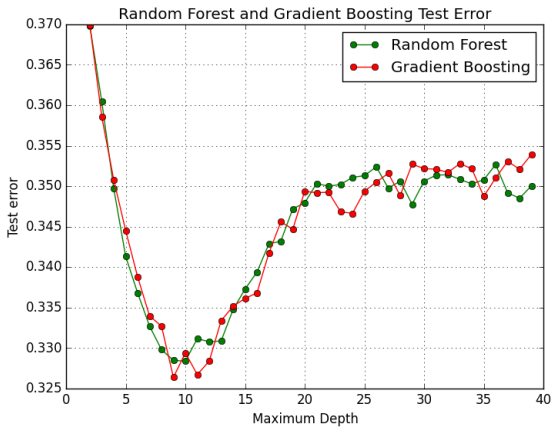

Figure 3: Sensitivity of Test Error with respect to the trees' maximum depth

optimum maximum depth for Random Forest is 10, whereas for Gradient Boosting is 9. Using these parameters, the test error in Random Forest was 0.329, and the lowest test error in Gradient Boosting was 0.326.

Note that the major difference between the 2 types of algorithm is that during the training process, Random Forests are trained with random samples of the data exploiting the fact that randomization have better generalization performance. On the other spectrum, the Gradient Boosting Algorithm tries to add new trees to complement the ones already built. It also tries to find the optimal linear combination of trees (assume final model is the weighted sum of predictions of individual trees) in relation to a given train data. This extra tuning might be deemed as the difference. Note that, there are many variations of those algorithms as well. Within the scope of the project we have used the most common version of the algorithms as described above.

Learning Curves

Considering that the previous analyses indicated that the best algorithms to predict recidivism are Random Forest and Gradient Boosting, our team looked for improving the performance of these algorithms. The parameters found in the **Parameter Estimation** Subsetcion were used. The performance was measured by the test error reported in a 5-fold cross validation. To diagnose these algorithms, i.e. to verify whether or not the test error could be reduced and to find possible ways of reducing it, our team constructed learning curves. These curves compare how the training error and the test error vary as a function of the size of the training sample.

Figure 4 shows the learning curves for Random Forest and Gradient Boosting algorithms. Additionally, it shows how the simple Logistic Regression algorithm compares to

both the Random Forest and Gradient Boosting algorithms. This figure indicates that the Logistic Regression algorithm has its test error very similar in value to its training error. This may explain that in order to improve our prediction, there is a need for reducing the bias of the problem. Therefore, looking for additional features could improve our predictions. Conversely, the Random Forest algorithm shows that its training and test error are very dissimilar. This fact might be associated to a model with high variance. Nevertheless, after reducing the variance of the model by modifying the maximum depth and the minimum number of leaves in the model, no better test errors were found. The Gradient Boosting method situates between both previous methods. Its training and test error are not as similar as in the Logistic regression, but not as dissimilar in the Random Forest. Interestingly, this method achieves the lowest test error: 0.326.

Remarkably, all the methods have a flat test error curve when using 6000 data points (nearly a third of the data set) or more. This lead us to think that the Machine Learning algorithms reached converged values, and therefore a larger data set would not improve our predictions.

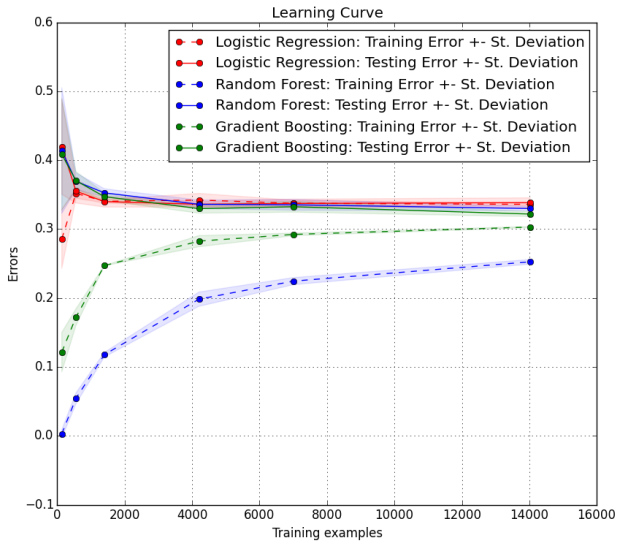

Figure 4: Learning Curves

Feature Engineering

Statistical Approach

One of the core of the study was exploring the impact of the different features in predicting if an individual is likely to go back to jail after his release. A first exercise that we did was looking at the distribution of our features within the data-set used. Note that there is only 5 non-binary

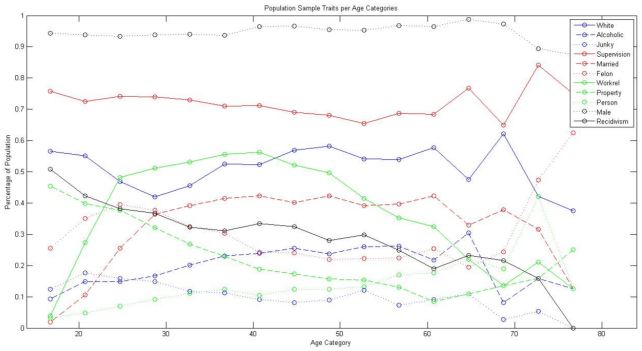

Figure 5: Distribution (%) of the features per age groups

features: age, time served, number of school years, number rules violated in prison and number of priors. We have run different simple statistical visualization methods such as histograms, distribution of the binary features given segments of population (per age, time served and school years) and finally mapping the distribution of a binary feature at the intersection of 2 segments.

As the plot above shows, we can easily catch obvious trends such as the fact that gender is unlikely to be a good predictor given the proportion of male in our population. Also immediate patterns are visible marriage and age: youngsters and elderly have lower companionship rates. Mainly, this analysis led us to think that we did not necessarily need to take into account all the features to make a good prediction.

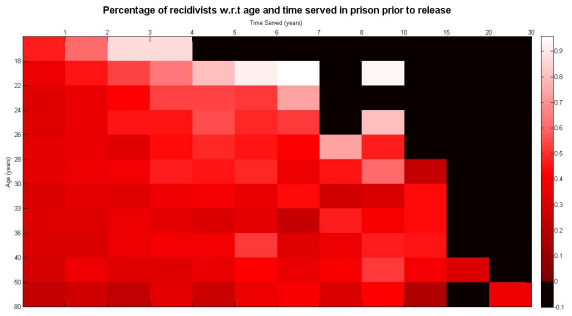

Figure 6: Mapping of recidivist per groups of age and time served in prison

Figure 6 and 7 show matrices with age bins along the Y-axis and respectively, School Years and Time Served in the X-axis. The whiter the rectangle is, the more frequent that person goes back to jail. These graph indicates a strong correlation between age and years spend in prison when looking at the recidivist population. This initial exploration lead us to manually (using linear logic combinations).

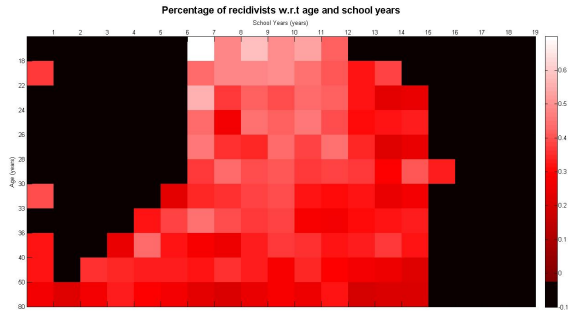

Figure 7: Mapping of recidivist per groups of age and time served

Feature Selection

We have used both forward backward (Figure 8) feature selection to measure the importance of the features w.r.t to one another. Without doubts the most crucial features are (in order):

(1) Ethnicity
(12) Time Served
(14) School years
(15) Rule violations

Part of the difficulty was understanding the which features were the most indicative of individuals likely to recidive. Manually engineering features (linear logic combinations) as been explored un-fruitfully. We believe this approach should be pursued in feature work.

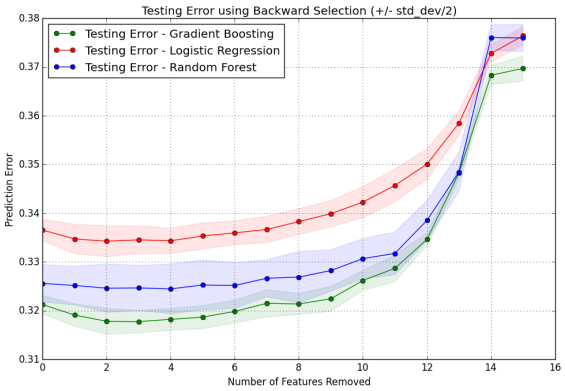

Figure 8: Backward Selection using LR, RF and GB

Limits and further work

Do race matter?

Extreme racial disproportionalities exist in American jail population. Therefore, it induces a bias in our model. The question is to know weather to include the feature Race or not. In an optic to make our model as fair as possible it would be interesting to try and remove the feature Gottfredson (1996).

Online learning

People's behavior and trends are always evolving in a society. Therefore if such an algorithm is used for judicial decision, it would be important to constantly keep updating it as we get new data points. Consequently it is proposed to use online learning algorithms.

Ethic

The problem we are trying to solve raises a lot of ethical questions. How good must predictive efforts be to justify using them to take restrictive actions that implicates the liberties of others? This is a very ethical concern that needs to be thought through in the case where the algorithm is used for real decision making.

Conclusion

Our work is an attempt to recidivism modelling. We use features that are easily accessible by the judge and have a significant impact on the probability of recidivism. It was determined that the best predictive model is the gradient boosting algorithm using 13 features (follow, felony property) with an error of 31.8%. In further work, this error rate could be significantly decreased by using a bigger data set. Huge data set with millions of points have already been collected in the US. Yet, we could not access it for this project since an IRB protocol is required for sensitive data on human subjects. This exploration is not be confused with a willingness to substitute judge by machines. On the contrary, it helps them make better decision to improve the American criminal justice system, to make it more just, objective and fair.

References

Gottfredson, S. (1996). Race, gender, and guidelines-based decision making. *Journal of Research in Crime and Delinquency*, 33(1):49–69.

Milgram (2014). Why smart statistics are the key to fighting crime. *Ted Talk*.