1 Data

The data that will be used by the machine learning algorithm to build a prediction model is data from Broward County, Florida (same county from which ProPublica gathered data to analyze the COMPAS algorithm) on criminal recidivism—that is whether a criminal will commit another crime in the future.

Each row in the data (i.e., "data instance" or just "instance" for short) contains information on one individual who was charged with a crime in Broward County as some point in the past (during a particular window of time). Each data instance contains a number of input features (described below) for that individual and also contains an output value that indicates if that individual went on to recidivate (commit another crime) in some time window in the future. Based on this data, the machine learning algorithm will learn a model (i.e., prediction function) that tries to predict if an individual (given their input features) will recidivate.

Input features

The original input features for each individual are as follows:

- Juvenile felony count: A count of the number of felony convictions this individual has as a minor (juvenile). Originally, this feature was simply an integer value, but for this assignment it was transformed into a feature with four categorical values representing ranges/bins of counts. Those bins are:
  - Count = 0
  - Count = 1
  - Count = 2
  - Count >= 3
- Juvenile misdemeanor count: A count of the number of misdemeanor convictions this individual has as a minor (juvenile). As with juvenile felony count above, this feature was originally an integer value, but for this assignment it was transformed into a feature with four categorical values representing the same ranges/bins of counts as above (namely: 0, 1, 2, >=3).
- Juvenile "other" count: A count of the number of non-felony/non-misdemeanor convictions this individual has as a minor (juvenile). Such "other" convictions are less severe than felonies and misdemeanors (i.e., infractions). This feature was also originally an integer value, but was transformed into a feature with four categorical values representing the same ranges/bins of counts as above (namely: 0, 1, 2, >=3).
- Prior convictions count: A count of the total number of prior convictions this individual has had as an adult. This feature was also originally an integer value, but was transformed into a feature with four categorical values representing the same ranges/bins of counts as above (namely: 0, 1, 2, >=3).
- Degree of charge: The degree of the current charge that this individual is facing. The only possible values for this feature are: "felony" or "misdemeanor".
- Description of charge: The type of crime with which the individual is being charged. Originally, this feature had over 400 possible values, but they were consolidated into the following 12 high-level categories. A criminal charge only falls into one category:
  - No charge
  - License issue

  - Public disturbance
  - Negligence
  - Drug-related
  - Alcohol-related
  - Weapons-related
  - Evading arrest
  - Nonviolent harm (i.e. stalking, tampering with victim, property damage, etc.)
  - Theft/fraud/burglary
  - Lewdness/prostitution
  - Violent crimes

- Age: The individual's age at the time of arrest. Originally, this feature was simply an integer value, but for this assignment it was transformed into a feature with three categorical values representing ranges of ages (to match the same age bins used in the ProPublica analysis for this feature). Those age ranges/bins are:
  - Less than 25 years old
  - 25 to 45 years old
  - Greater than 45 years old
- Gender: The individual's gender. This feature only had two values in the data (female and male).
- Race: The individual's race. This feature has six values:
  - Other (i.e., none of the races below)
  - Asian
  - Native American
  - Caucasian (same as "White" in the ProPublica analysis)
  - Hispanic
  - African-American (same as "Black" in the ProPublica analysis)

"One-hot" feature encoding

To simplify the machine learning process and the analysis of the results, all the data was encoded using a "one-hot" feature encoding. A one-hot encoding simply takes an input feature with n discrete values and replaces it with n binary features (i.e., features the only either have the value 0 or 1), where only one of those n features has value 1 (corresponding to the actual value of the underlying variable) and the other n – 1 features have value 0. More concretely, consider the binned version of Age with 3 distinct values. Rather than having an Age feature with values 1, 2, or 3 (corresponding to the bins: "Less than 25 years old", "25 to 45 years old", and "Greater than 45 years old"), a one-hot encoding would instead have three (binary) features as follows:

- Age is less than 25
- Age is 25 to 45
- Age is greater than 45

The data row representing an individual would then include three binary values corresponding, respectively, to these age-based features, where only one of the three age-based features would have value 1 (the other two would have value 0), depending on which age range the individual was in. So, an individual who was 21 years old would have their age represented by the series of values: 1, 0, 0, since their age in the first bin. A 30-year-old would have their age represented by the series of values: 0, 1, 0, since their age is in the second bin. And a 52-year-old would have their age represented by the series of values: 0, 0, 1, since their age is in the third bin. Such an encoding allows two benefits. First, it allows for different weights to be learned for each different age ranges (or, more generally, different values of

any underlying feature) since each feature value/range is transformed into a separate feature. Second, it allows for analysis of specific subpopulations more easily by just examining data where a particular feature has value 1 (i.e., looking at the results for just a specific age range or just a specific race) to compare subpopulations more directly.

Data file format

As mentioned above, the data files contain one row (line) per individual in the data. The rows are comma-separated values. Each row contains the one-hot encoding of all the input feature values for that individual as well as the output value (if they recidivated (value 1) or not (value 0)). The output value is always the last value in the row. Thus, each row has 42 values: 41 binary input features representing one-hot encodings of the attributes of an individual and a binary value indicating if the individual recidivated or not. In the file constants.py you will find a set of constant values (essentially, an enumeration) that lists all the indexes of all input features in the data to make it easier to programmatically refer to particular input features.

Training and Testing data

The data is split into two files. There is a "training" data file (named `recidivism-trainingdata.csv`) which is used to train the machine learning algorithm (i.e., determine the model weights). The "testing" data file (named `recidivism-testing-data.csv`) is then used to determine the accuracy of the model after the training phase is complete. In other words, when we describe training a model below, you should take that to mean that the algorithm is working only with the training data to determine the weights in the model. When we then describe testing a model you should take that to mean that only the testing data (which is distinct from the training data) is used to determine how well a model does at making predictions. The model weights are not updated when using the model is making predictions on new (testing) data.

Dataset class

The file `dataset.py` implements a simple Dataset object that reads the data files described above and stores them in an object that allows access to the rows in the data. Please review the file `dataset.py` (which is thoroughly commented) for more details.

2 Code

The Peceptron machine learning algorithm

The machine learning algorithm used in this assignment is the Batch Perceptron Pocket algorithm. The details of how this algorithm works are described in the handout on "Probability and Machine Learning" available from the class website (and listed as one of the readings for class). This algorithm is implemented in the file `perceptronmodel.py` in the project that we provide. Note that a PerceptronModel can be created by giving it a Dataset object to train on. Importantly, the PerceptronModel is also instrumented so that it can be trained on just a subset of the input features in the data (rather than using all the features). This allows for comparing different models based on which input features they actually use (more on that below). Learned models (i.e., a set of weights) can also be saved to and read from files for easy comparison and storage. Note that you can also manually change a weight in a saved model file and then read it in to see how changing the weight impacts prediction results. See the file `perceptronmodel.py` for more details.

The main program

The file `algorithmicdecisionmaking.py` is the main program file that does the work of creating a Perceptron model using the training data and then reporting the prediction results of the model on both

the training and testing data. Note the function `select_features_to_use` in this file, which allows you to select a subset of the input features to train the model. In fact, the initial version of the algorithm does not use all the available input features—you will see some of them commented out in the list of features to use—to train the model. Input features that are not used while training the model will have associated weight values of 0 (which are never updated during training), and thus will neither impact the training process nor predictions of the algorithm. There is also a function named `print_results` in this file that reports various statistics related to the performance of a particular model (PerceptronModel) on a data set. You can either print the resulting statistics for the entire data set or just that subpopulation of the dataset that has some particular value for a feature (e.g., print the results only for those individuals who have a 1 for the `Age_Less_Than_25` feature).

Acknowledgments This assignment was written and designed by Rob Reich, Jeremy Weinstein, and Mehran Sahami, all at the Stanford University for their CS182 course. Student Michael Dworsky also categorized the recidivism data and developed an early prototype for the assignment. They have given me permission to use their materials.
Any changes made have been minor with the aim of providing context, or adapting to the specifics of CSC-395 at Grinnell College.