CS182 January 14, 2026
Mehran Sahami

Assignment #1: Analyzing Algorithmic Decision-Making
Due: 11:59pm (Pacific Time) on Tuesday, January 27th

Many thanks to Michael Dworsky for categorizing the recidivism data and developing an early prototype for this assignment.

In exploring algorithmic decision-making, especially in the context of criminal recidivism prediction, we want you to have first-hand experience in interrogating the model built by a machine learning algorithm. Your investigation includes not only being able to transparently see the code for the machine learning algorithm (which is what some lawmakers argue is a necessary safeguard in deploying algorithms for such decision-making), but also assessing the "fairness" of the model produced by the algorithm on a number of criteria while thinking about the limitations of the data given. Ultimately, we want you to determine what you would do to make the decision-making model produced by the algorithm more "fair," and justify this position in a short memo to a company executive.

Getting set up with Python

The code for this assignment will be provided in Python. You will need to install Python/PyCharm to use Python on your computer. Instructions for software installation are given on the "Software" page of the CS182 website. Please follow the instructions carefully to download and install PyCharm. Once you have installed PyCharm, you should refer to the handout "Using PyCharm in CS182" (available on the "Handouts" webpage of the CS182 website) to get more information on downloading and running the project for this assignment and how to run the project.

The assignment

As mention previously, your task in this assignment is to evaluate a decision-making algorithm. This algorithm uses machine learning to learn a decision-making model for criminal recidivism prediction. We will start by describing the data that is used by the algorithm and then discuss the algorithm itself. Finally, we will explain the assignment itself—that is, what you are required to as part of your evaluation of the algorithm.

The 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-training-data.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.

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 **`algorithmicdecisionmaker.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).

What you need to do

Say you are employed at a company named *JudgeSoft* that is producing an algorithm that will train a model to make a prediction as to whether an individual charged with a crime is likely to recidivate or not. The predictions of the model learned by the algorithm will be used by Broward County judges as one of many factors in determining whether to *release* an individual charged with a crime until the time of their trial (i.e., grant them bail) or keep them *detained* (i.e., deny bail) until the trial. Thus, whether the individual is likely to commit another crime (i.e., recidivate) will impact the judge's decision.

Having been just recently hired by *JudgeSoft* you find that a machine learning algorithm has already been developed for this project. The company Chief Technology Office (knowing that you are interested in ethics and technology after seeing CS182 on your transcript) asks you to investigate the algorithm, answer some questions about it, and make some recommendations before a final product is shipped. More specifically, she wants you to answer the questions below. You should write up the answer to all these questions in a **single PDF file** titled "**`Writeup`**". Details on how to submit your project folder is provided at the end of this handout.

Questions to answer

1. Without running the program (or, at least, not considering the results if you did run the program), review all the code files in the **`AlgorithmiDecisionMaking`** project to understand how the overall program works.

    a. (2 points) Note that the programmers of the algorithm had heard about "protected" characteristics, such as age, gender, and race, and that impacted their code. Do you believe that the algorithm itself (as coded) includes any biases? Briefly (in a paragraph or two) explain your answer. Include references to concepts from class as needed to justify your answer.

b. (2 points) Again, prior to running the algorithm (or, at least, not considering the results if you did run the program), consider the case where the algorithm was coded to include the use of the input features age, gender, and race during the learning process. Would you believe that in this case the algorithm includes any biases? Briefly (in a paragraph or two) explain your answer. Include references to concepts from class as needed to justify your answer.

2. (6 points) Although there are six race-based input features, for this part of your investigation you can focus on just African-Americans and Caucasians as was done in the ProPublica investigation. Considering the original model (which does not use any protected features), train a model using the training data. Then, using this learned model, compute the statistical results (use the **`print_results`** function and any other statistics you believe are relevant that you want to add to the code) for African-Americans and Caucasians on the training data and the testing data. Report the statistics you computed. Comment on how the various results you obtained from the model compares with ProPublica's analysis of the differences between predictions for African-Americans and Caucasians in the COMPAS algorithm. Explain the similarities/differences you see.

3. (6 points) Based on the results you obtained from question 2 above, do you believe that the model learned by the algorithm is "biased"? Explain why or why not. Justify your answer with respect to *at least three* of the notions of anti-classification, classification parity, calibration, and disparate impact. Use quantitative results as appropriate to make your argument.

4. (6 points) Modify the code (**`select_features_to_use`** function) to include all the input features for age, gender, and race (along with the other input features already included in the model) and train a new model with these input features on the training data. Based on the results you obtained with the new model, do you believe that this new model is more or less "fair" than the old model (built in question 2) which did not use the protected characteristics? Explain your decision, including how you define the notion of "fair". Justify your answer with respect to *at least three* of the notions of anti-classification, classification parity, calibration, and disparate impact on the data. Use quantitative data as appropriate to make your argument. Note: you might want to consider protected characteristics beyond race, such as gender and age, in your answer.

5. (18 points) Modify the code however you like, including (but, not limited to):
   - Choice of input features to include while training the model
   - Making adjustments to the weights, including (automated or manual) adjustment to any of the weights to essentially create differential prediction thresholds for different subpopulations. (For example, increasing the weight for a particular input feature makes individuals with that feature more likely to be classified as positive (prediction value 1) by the model, as their overall weighted sum will increase as a result).
   - Whatever else you would like to do (e.g., training different models for different subpopulations, other means for setting different prediction thresholds based on

various features, changing the number of epochs for training, interrogating the data set, researching the potential provenance of data from the criminal justice system, etc.)

Your goal in making these modifications is to make the decision-making algorithm (the resulting learned model) as "fair" as possible in your assessment. Write an approximately 500 word (~2 page) memo to the company CTO about your work. (Note that statistics and their labels/descriptions, as well as tables and diagrams you might want to include, do not count as part of the ~500 words of text.) You can assume the CTO has a solid technical background—she knows how to code well, understands what machine learning is and how it works, and has done all the readings and attended all the classes in CS182, so she knows about different concepts of fairness, and understands machine learning and statistics. Your memo should explain the final model you have come up with in detail (e.g., choice of features, other adjustments you would make, etc.) and explain why you believe these choices have made the model as "fair" as possible (especially with respect to the various "fairness" criteria you are aware of), while also noting potential limitations. Try to be as clear as possible with regard to your definition of "fair". Use quantitative measurements and qualitative arguments to justify your claims.

Note: this part of the assignment is not an assessment of how much you know about machine learning. For example, building a complicated model for which you have difficulty justifying its "fairness" will not impress the CTO to whom you are writing this memo. The goal here is to make an argument that shows your assessment of a model's "fairness" in a real-world setting, where you have control over what that model (and the algorithm that built it) is.

Submitting your work on Gradescope

If you have not already done so in the past, you should sign-up for a Gradescope account. Go to the website: **`https://www.gradescope.com/`**. If you already have an account, you should login. If you don't have an account, then click the "Sign Up" button and follow the instructions to create an account. Once you are logged in to your Gradescope account. You should enroll in CS182, by navigating to your Account Dashboard by clicking the Gradescope logo in the top left, and click "Enroll in Course" in the bottom right corner of the screen. You will then be asked for a course Entry Code. The Entry Code for CS182 this Winter is: **6XXZ77**

If you need help with Gradescope, you can get more information at:
`https://www.gradescope.com/help#help-center-section-student-workflow`

This assignment will be submitted in two parts on Gradescope. Please make sure to **submit both parts of the assignment**, as described below.

Submitting Part 1 (question write-ups):
As mentioned previously, you should write up the answer to the questions above in a **single PDF file** titled "Writeup". You should submit your Writeup file on Gradescope as a submission to "Technical Assignment #1 - Writeup". You should get a confirmation from Gradescope once

your file is successfully uploaded. Make sure to maintain a backup copy of your Writeup file in case there are any issues with your submission on Gradescope.

Submitting Part 2 (code):
Your **`AlgorithmicDecisionMaking`** project folder should contain the final version of the code changes you made with respect to question #5—that is, running your code should generate what you believe is the most "fair" model that you can construct from the training data, and then print the results of this model on the training and testing data, separately. Also, include any other changes you made (such as additional statistics you may have computed from the model). Make sure that the code you submit matches the model you describe in your written memo to the CTO in question 5. You should then create a ZIP file of the **`AlgorithmicDecisionMaking`** project folder and name the resulting ZIP file with your first and last name (e.g., **`PatJones.zip`**).

Upload your ZIP file to Gradescope as a submission to "Technical Assignment #1 - Code". You should get a confirmation from Gradescope once your file is successfully uploaded. Make sure to maintain a backup copy of your project files in case there are any issues with your submission on Gradescope.