CSE474 Introduction to Machine Learning
Programming Assignment 3
Probabilistic Methods - Understanding Bias in Machine Learning

Due Date: May 8th 2019

1 Introduction

This programming assignment has two parts. In the first part, you will implement a *Naive Bayes Classifier* and test it on a publicly available data set. In the second part, you will manipulate the data characteristics to understand how classifiers get impacted by the underlying bias in the training data.

After completing this assignment, you should be able to understand:

- How a Naive Bayes classifier works and how to incorporate prior information into the model?
- How bias in the training data could lead to a biased and unfair machine learning model? Within this topic you will learn:
    - How to measure bias (or fairness) of an algorithm with respect to bias-sensitive information?

To get started with the exercise, you will need to download the supporting files.

1.1 Files included in this assignment

- *german.data.pickle*: The data set file. This dataset hosted & provided by the UCI Machine Learning Repository[1] contains mock credit application data of customers. Based on the features provided in the dataset, the customers are classified as good or bad and the labels will influence credit approval. The dataset contains several features such as: - Credit History - Status of Bank Accounts - Employment History and several others as would pertain to a credit application. The data file that is provided is a processed version of the original data set. We have discretized some of the numeric features and mapped the codes to integers. The pickle file has three numpy arrays:
    1. *features*: A 1000×18 numpy array representing the 18 features for 1000 customers.
    2. *sensitive*: A 1000×2 numpy array representing 2 sensitive features for each customer, viz., gender and age. These will not be used as features in the classifier, but will be used to study the *implicit* bias in the classification of customers into good or bad credit rating.
    3. *labels*: A 1000×1 numpy array containing the target labels (1 - good rating, 2 - bad rating).
- *data_info.txt*: The meta data file that contains the descriptions for the features.
- *nbFunctions.py*: Files containing Python functions for this programming project. Contains function definitions -
    - *nbTrain()*: trains the Naive Bayes Classifier, given the training data, training labels, and prior information. *You need to make changes to this function.*

[1] `https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data)`

  - *nbPredict()*: predicts the label of test data given the trained model. *You need to make changes to this function.*

- *nbScript.py*: Python script for this programming project. This script calls the implemented functions in *nbFunctions.py*. *You will need to change the data characteristics and run this script many times.*

2 Your Tasks

- Implement **the Naive Bayes Classifier** (training and prediction) that utilizes priors on the target class variable as well as individual features (See Section 3).
- Evaluate your implementation on the *credit card data set* and report results (See Section 4).
- Measure the *classifier bias* in the results using the sensitive attributes (age and gender) (See Section 5).
- Bias the training data using the sensitive attributes, and measure the impact on the classifier bias (See Section 6).
- Write a report to explain the results.

3 Naive Bayes Classification

As discussed in class, in a Naive Bayes model, the input features and the target variable are treated as random variables, $\mathbf{X}$ and Y, respectively, where $\mathbf{X}$ is a vector consisting of D independent random variables, i.e.,

$$P(\mathbf{X}=\mathbf{x}) = \prod_{j=1}^{D} P(X_j = x_j) \tag{1}$$

$$= \prod_{j=1}^{D} p(x_j) \tag{2}$$

where $p(x_j)$ is the probability mass or density function, depending on if each random variable is categorical or continuous, respectively. For this assignment, each feature is categorical, i.e., it takes one of out K_j possible values, and hence the corresponding random variable will be treated as either a *Bernoulli* random variable (if $K_j = 2$) or *Multinoulli* random variable (if $K_j > 2$).

For this assignment, the target variable, Y, can take only two values (1 - good credit or 2 - bad credit). So we will treat Y as a Bernoulli random variable.

To predict the class for a data point ($\mathbf{x}$), the Naive Bayes Classification equations are:

$$P(Y=1|\mathbf{X}=\mathbf{x}) = \frac{P(Y=1)\prod_{j=1} P(X_j = x_j|Y=1)}{P(Y=1)\prod_{j=1} P(X_j = x_j|Y=1) + P(Y=2)\prod_{j=1} P(X_j = x_j|Y=2)} \tag{3}$$

$$P(Y=2|\mathbf{X}=\mathbf{x}) = \frac{P(Y=2)\prod_{j=1} P(X_j = x_j|Y=2)}{P(Y=1)\prod_{j=1} P(X_j = x_j|Y=1) + P(Y=2)\prod_{j=1} P(X_j = x_j|Y=2)} \tag{4}$$

The class probabilities, $P(Y=1)$ and $P(Y=2)$ can be calculated as:

$$P(Y=1) = \hat{\theta}_{Bayes} = \frac{N_1 + a}{N + a + b} \tag{5}$$

$$P(Y=2) = 1 - \hat{\theta}_{Bayes} = \frac{N_2 + a}{N + a + b} \tag{6}$$

where N is the total number of training examples, and N_1 and N_2 are the number of training examples when $Y = 1$ and $Y = 2$, respectively. We are assuming a *Beta* prior for Y and using the *Bayesian Averaging* estimate of the posterior, which is same as the expected value of the *Beta* posterior ($Beta(a+N_1, b+N_2)$, where $N_2 = N - N_1$). a and b are the parameters of the *Beta* prior for the class variable Y.

The class conditional probabilities for each feature, i.e., $P(X_j = x_j|Y = 1)$ and $P(X_j = x_j|Y = 2)$, can be similarly computed. However, note that since each X_j can take more than 2 values, we cannot model it as a *Bernoulli* random variable. Instead, we will have to model it as a *Multinoulli* random variable. The conjugate prior for a *Multinoulli* distribution is a *Dirichlet Distribution*. You will notice that the (*Multinoulli*, *Dirichlet*) pair is just a generalization of the (*Bernoulli*, *Beta*) pair.

- A *Bernoulli* random variable takes one out of 2 possible values, while a *Multinoulli* random variable takes one of out K possible values.
- A *Beta* distribution sample is a value between 0 and 1. A *Dirichlet* distribution sample is a K length vector, where each entry is between 0 and 1.
- The parameters of a *Beta* distribution are two numbers, a and b. The parameters of a *Dirichlet* distribution is a vector of K numbers, generally denoted as $\alpha_1, \alpha_2, \ldots, \alpha_K$.
- The posterior for a *Beta* prior ($\sim Beta(a, b)$), when the likelihoood is *Bernoulli* (counts - (N_1, N_2)), is also a *Beta* distribution ($\sim Beta(N_1+a, N_2+a)$). The posterior for a *Dirichlet* prior ($\sim Dirichlet(\alpha_1, \alpha_2, \ldots, \alpha_K)$), when the likelihoood is *Multinoulli* (counts - $(N_1, N_2, \ldots, N_K)$), is also a *Dirichlet* distribution ($\sim Dirichlet(\alpha_1 + N_1, \alpha_2 + N_2, \ldots, \alpha_K + N_k)$).

Which means that the class conditional probabilities (using the Bayesian posterior), can be expressed as:

$$P(X_j = x_j|Y = 1) = \hat{\theta}_{1j} = \frac{N_{1j} + \alpha_{1x_j}}{N_1 + \sum_{k=1}^{K_j} \alpha_{1k}} \tag{7}$$

where N_{1j} is the number of training examples where $Y = 1$, for which the j^{th} feature takes the value x_j ($x_j \in \{1, \ldots, K_j\}$), and α_{1j} is the j^{th} entry of the parameter vector of the *Dirichlet* prior for the feature j. Typically, we assume that the prior is symmetric, that is all entries in the parameter vector are same ($\alpha_{1k} = \alpha, \forall k$). Moreover, we use the same α for both classes. Thus, the class conditional probability expressions become simpler:

$$P(X_j = x_j|Y = 1) = \hat{\theta}_{1j} = \frac{N_{1j} + \alpha}{N_1 + K_j\alpha} \tag{8}$$

Here K_j refers to the number of possible values that the j^{th} feature can take. The expression for $Y = 2$ will be:

$$P(X_j = x_j|Y = 2) = \hat{\theta}_{2j} = \frac{N_{2j} + \alpha}{N_2 + K_j\alpha} \tag{9}$$

Training

In the training step, you will have to compute the quantities $\hat{\theta}_{Bayes}$ (See (5)), $\hat{\theta}_{1j}$ (See (8)) and $\hat{\theta}_{2j}$ (See (9)).

Testing

In the testing step, you will have to compute the probabilities of a test instance to belong to class 1 or 2, using the expressions in (3) and (4), and output the class with higher probability.

4 Evaluation

We will use *cross-validation* to measure the performance of the Naive Bayes classifier on the provided data set. The idea behind cross-validation is to split the given data set into training and test data sets using a random splitting. However, this splitting is done k times (also referred to as k-fold cross-validation), ensuring that each example in the data set occurs in the test data set for exactly one fold. We will be using the cross-validation module in the `sklearn` library.

In the `nbScript.py` file, you will notice that we are not using accuracy as the evaluation measure. This is because accuracy metric gets severly impacted by imbalance in the class distribution. For instance, if a data set has 700 instances of class 1 and 300 instances of class 2, and a classification algorithm assigns a label 1 to all of the instances, the accuracy of the algorithm will be 70%, which can be misleading. To account for class-imbalance there are other measures, such as the F-measure. The F-measure, for each class, is the harmonic mean of the *recall* (number of correctly classified test instances of that class divided by the actual number of test instances of that class the data set), and the *precision* (number of correctly classified test instances of that class divided by the total number of test instances classified as that class). We will be reporting the F-measure averaged over two classes.

5 Classification Bias

As the use of machine learning to make decisions about people has increased, so has the drive to make fairness-aware machine learning algorithms. A considerable body of research over the past ten years has produced algorithms for accurate yet fair decisions, under varying definitions of fair, for goals such as non-discriminatory hiring, risk assessment for sentencing guidance, and loan allocation.

In this assignment we want to study the impact of a biased training data on the fairness of the machine learning algorithm that learns from it. In particular, we want to test if the classifier, trained on the provided input data, could be unfair to individuals of a certain *gender* or *age group*. In the provided dataset, we have provided, in an array called *sensitive*, the gender and age information for each customer. We refer to these features as *sensitive* features. While, the sensitive features are not used as inputs to the learning algorithm, and hence are not directly used to assess if a customer has good or bad credit, *implicit bias* in the training data could result in the algorithm being unfair to a certain gender or an age group.

Measuring Fairness

There are many strategies to measure how unfair an algorithm is, with respect to a sensitive feature. Here we will use one measure used in the literature, called *Disparate Impact* (DI), which is computed as:

$$DI = \frac{P(\hat{Y} = 2|S \neq 1)}{P(\hat{Y} = 2|S = 1)} \tag{10}$$

Here $\hat{Y}$ denotes the predictions made by the classifier. S denotes a sensitive feature such that $S = 1$ refers to a "privileged" feature and $S = 2$ refers to the unprivileged feature. The DI measure compares the probability of an unprivileged customer to be assigned bad credit and the probability of a privileged customer to be assigned bad credit. Clearly, higher value for DI would mean that the algorithm is unfair to the unprivileged customers. Ideally the value of DI should be 1, which means that the probability of a customers to be assigned a bad score is independent of the value of the sensitive variable.

For this assignment you will experiment with two sensitive features - age and gender. For age, we want to check if the algorithm is unfair to younger (< 25) customers and for gender, we want to check if the algorithm is unfair to female customers. Measure the disparate impact for each of the two sensitive features and report the results.

6 Inducing Bias in Training Data

One of the reasons why a classifier might be unfair is because of the bias in the training data. In this part you will study the impact of the bias in the training data on the fairness of the final algorithm. You will use the provided function (`genBiasedSample`) which induces artifical bias in the data set by oversampling bad examples from the unprivileged set of cusomters. A parameter in the function is p which denotes the probability of picking up a "bad" customer from the unprivileged set. Clearly, if $p = 0.5$, the data set will have no artificial bias. For $p > 0.5$, the data will be biased against the unprivileged customers. For $p < 0.5$, the data will be biased against the privileged customers.

You will have to experiment with different values of p, $0 \leq p \leq 1$. For a certain value of p, you will have to get the resampled data set (using `genBiasedSample`), get the predictions from your Naive Bayes implementation (in cross validation mode), and measure the disparate impact. You will then plot the disparate impact against p and explain the results. Do this for both gender and age features.

7 Submission

You are required to submit a single file called *proj3.zip* using UBLearns.
File *proj3.zip* must contain 2 files: *report* and *code* (`nbFunctions.py`).

Using UBLearns Submission: Please continue using your programming groups. Contact the instructors if you need to change groups on UBLearns.

Project report: The hard-copy of report will be collected in class at due date. Your report should include the following:

- Report the cross-validation F-score for the implemented Naive Bayes classifier on the provided data set. Vary the prior probabilities (a, b,and α) and report the results.
- Report the *Disparate Impact* score of the cross-validation predictions of the Naive Bayes classifier (you can choose a specific value for the priors) for the two sensitive features - age and gender. Is your classifier unfair? State reasons why you think so.
- Using the provided resampling function, measure the disparate impact of the classifier as a function of p (probability of resampling a bad and unprivileged customer). Report your findings.

8 Grading scheme

The TAs will deploy a testing script that will test the functionality of individual functions that you submit within the *nbFunctions.py* file. Full points will be available if the accuracy and disparate impact values are within 5% of the values reported by our code.

- Successfully implement the Naive Bayes Classifier: 50 points (*fit()* [25 points], *predict()* [25 points]).
- Successfully implement the Disparate Impact measure: 20 points.
- Project report: 30 points
  - Explanation with supporting figures of the impact of priors on the performance of the classifier: 10 points
  - Explanation of the unfairness of the Naive Bayes algorithm with respect to the two sensitive features: 10 points
  - Explanation with supporting figures of the relationship between p and fairness of the classifier: 10 points

References

[1] Sorelle A. Friedler, Carlos Scheidegger, Suresh Venkatasubramanian, Sonam Choudhary, Evan P. Hamilton, and Derek Roth. 2019. A comparative study of fairness-enhancing interventions in machine learning. In Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT* '19). ACM, New York, NY, USA, 329-338. DOI: https://doi.org/10.1145/3287560.3287589