Today we are going to explore the methods of unfairness problem in Machine Learning. Since Propublica organization pointed out that the COMPAS (Correctional Offender Management Profiling for Alternative Sanction) system which is a database containing the criminal history are discrminatory against race and gender.It's analysis shows that Black defandants were often predicted to be at a higher risk of recidivism than they actually were while white defandants were predicted to be less risky. Recidivism is defined as defandants reoffend and get arrested again within two years. And the risk scores evaluated by their systemun results in unfairness between the defandant's recidivism situation.

In [4]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [5]:
```
%%capture
%run ../lib/LFR_model.ipynb
%run ../lib/DM_DM_sen_Model.ipynb
```

Data Preprocessing: From the ProPublica notebook, we removed the rows that

1. charge date of a defendants Compas scored crime was not within 30 days from when the person was arrested
2. the recidivist flag - is_recid == -1 if we could not find a compas case at all
3. those with a c_charge_degree of 'O' which means ordinary traffic offenses. It will not result in Jail time are removed
4. since we are only intersted in sample fairness between two races: African-American and Caucasian, we subsets our datasets

Here we introduce Learning Fair Representations techiniques to solve unfairness problem, the learning algorithm for fair classification is achieved by formulating fairness as optimization problem of finding good representation. The main idea in this model is to map each individual, represented as a data point in a given input space, to a probability distribution in a new representation space. General speaking, the goal of our model is to learn a good prototype set with the consideration of accuracy and statiscal parity.

Reference: Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, Cynthia Dwork, Learning Fair Representations, http://proceedings.mlr.press/v28/zemel13.html (http://proceedings.mlr.press/v28/zemel13.html)

Before we built the LFR model, we first transform each variable to a learnable indicator value. And according to the research paper, defandants with African-American race are regarded as non-sensitive group , Caucasian defandants are regarded as protected group.

As defined in the Learning Fair Representation paper, the Loss function **L = A_z * L_z+ A_x * L_x + A_y * L_y**, where **Ax, Ay, Az** are hyper-parameters governing the trade-off between the system's desire data, And relative **Lz, Lx, Ly** are defined as follows

The second term constrains the mapping to Z to be a good description of X. We quantify the amount of information lost in the new representation using a simple squared-error measure:

$$L_x = \sum_{n=1}^{N} (\mathbf{x}_n - \hat{\mathbf{x}}_n)^2 \tag{8}$$

where $\hat{\mathbf{x}}_n$ are the reconstructions of $\mathbf{x}_n$ from Z:

$$\hat{\mathbf{x}}_n = \sum_{k=1}^{K} M_{n,k} \mathbf{v}_k \tag{9}$$

The second term constrains the mapping to Z to be a good description of X. We quantify the amount of information lost in the new representation using a simple squared-error measure:

$$L_x = \sum_{n=1}^{N} (\mathbf{x}_n - \hat{\mathbf{x}}_n)^2 \tag{8}$$

where $\hat{\mathbf{x}}_n$ are the reconstructions of $\mathbf{x}_n$ from Z:

$$\hat{\mathbf{x}}_n = \sum_{k=1}^{K} M_{n,k} \mathbf{v}_k \tag{9}$$

The final term requires that the prediction of y is as accurate as possible:

$$L_y = \sum_{n=1}^{N} -y_n \log \hat{y}_n - (1 - y_n) \log(1 - \hat{y}_n) \qquad (10)$$

Here $\hat{y}_n$ is the prediction for y_n, based on marginalizing over each prototype's prediction for Y, weighted by their respective probabilities $P(Z = k|\mathbf{x}_n)$:

$$\hat{y}_n = \sum_{k=1}^{K} M_{n,k} w_k \qquad (11)$$

Therefore we defined the following function to calculate the relative value. And here we use scipy.optimizeo package to minimize our Loss function.

We split the protected group and unprotected group first and concatenate them together. Traning sets and testing sets are split proportionally as 6:1, you can see how does each defandants variables are being rescaled and manipulated

The results of the LFR model versus a logistic regression are:

In [6]:
```
1 return_lfr_accuracy()
```

```
the overall test accuracy for LFR is: 47.54%
the test accuracy for LFR for sensitive: 48.1%
the test accuracy for LFR for nonsensitive: 47.17%
the test accuracy for logistic regression is: 67.09%
the test accuracy for logistic regression for sensitive is: 64.56%
the test accuracy for logistic regression for nonsensitive is: 64.5700000
0000001%
```

In [7]:
```
1 return_dm_accuracy()
```

```
== Unconstrained (original) classifier ==


Accuracy: 0.649
||  s  || FPR. || FNR. ||
||  0  || 0.21 || 0.49 ||
||  1  || 0.25 || 0.48 ||



== Constraints on FPR ==


Accuracy: 0.649
||  s  || FPR. || FNR. ||
||  0  || 0.21 || 0.49 ||
||  1  || 0.25 || 0.48 ||
```

TO DO - add explanation for accuracies

In [10]:
```
1 !jupyter nbconvert --to pdf main.ipynb
```

In [ ]:
```
1
```