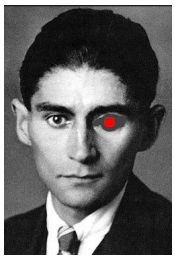
Kafkanator
A Fairness Toolkit and Optimization API for Data Science

Jose Francisco Saray Villamizar
Senior Data Scientist
jsaray@gmail.com

May 7, 2025



Motivation

Kafkanator API was developed with the purpose to help data scientists in some algorithmic fairness dimensions that have not yet received the attention they deserve. Mainstream algorithmic fairness discussions are about ML statistical predictors, and the main goal is to reduce bias in a prediction. However, there are many others fairness requirements that should interest a business. For example, for the ouput of a given in-house algorithm, how fair it is ?: Is my algorithm *preserving equity* in ressource assignation ?, is it *maximizing the ressource allocation* for each member of the population influenced by my algorithm? Is it assigning fairly ressources in a population taking into account their differences ?.

The next sections shows some simple daily life examples, and how you can use Kafkanator to evaluate how fair your business is doing it.

Fairness is a value that companies must respect and optimize. Customers have a fairness sense, and when they see an unfair event, they question about hiring a service or buying a product again from an unfair company.

1 The airplane example

I introduce you the atlantic flight Bogotá - Europe, carry out by the company coolflights.com.

A common airplane layout can be shown on diagram 1. It usually has N classes using *Confort* as the ressource to be shared between population. In diagram 1 , a seat labeled with 3 means it is more confortable than a seat labeled with 1 (much more space, for your legs, arms, better tv quality), and in consequence is more expensive.

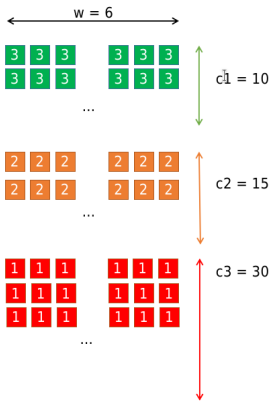


Fig 1. Bogota Europe flight seating by passenger classes

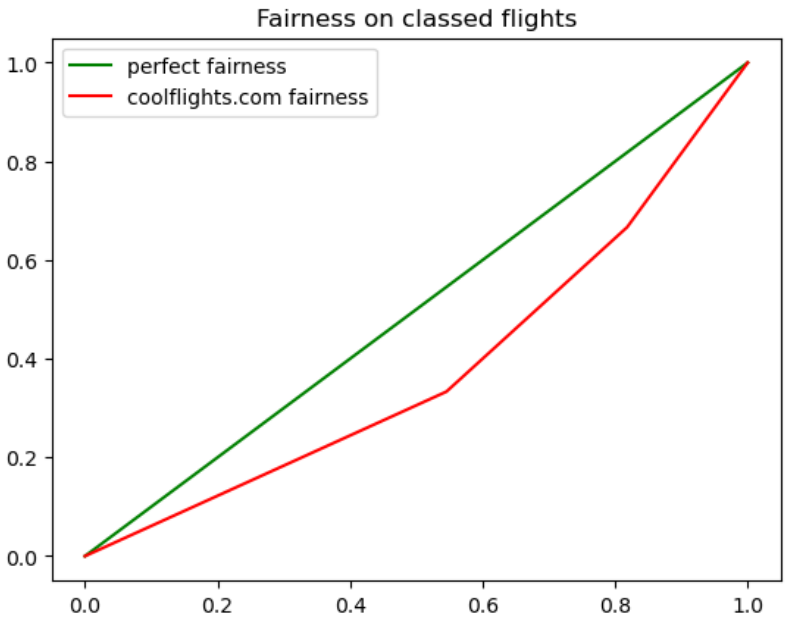
The interesting question is: how good is being shared *Confort* ressource between the passanger population in a N-class flight ?.The code presented here can be found in the notebook *tutorials / Airplane Confort Fairness Computing.ipynb*. I will provide here some explanations about that code.

1.1 Lorentz Curves

```
1  from kafkanator import lorentz_curve,gini
2  import matplotlib.pyplot as pyplt
3  import numpy as np
4
5  # Constants declaration from figure 1.
6  W = 6
7  POP_1 = 10
8  POP_2 = 15
9  POP_3 = 30
10 CONFORT_1 = 3
11 CONFORT_2 = 2
12 CONFORT_3 = 1
13 # Build two arrays of population and confort in increasing order:
14 # from the lower ressource population to higher ressource.
15 population = [ POP_3*W , POP_2*W, POP_1*W ]
16 confort = [ CONFORT_3*POP_3*W , CONFORT_2*POP_2*W, CONFORT_1*POP_1*W ]
17 # Obtain x and y coordinates to build the lorents curve in matplotlib.
18 (lorentz_x_coordinates, lorentz_y_coordinates) = lorentz_curve ( population , confort
   ↪  )
19
```

Code above calls the *lorentz_curve* Kafkanator method (line 18). Lorentz Curves [**1**] are a well known mathematical method to compute Inequity.One of the goals of Kafkanator is to gather inequity measures, and made them available for programmers and Data Scientists. Please see Kafkanator documentation to check the currently implemented inequity measures.

Once you have the x,y coordinates stored in *lorentz_x_coordinates* and *lorentz_y_coordinates* you can give them to matplotlib to build your curve, you will get something like this :



From this plot you can get valuable insights about fairness in that flight seating architecture: almost 60% of the passengers has less than 40% of the flight confort. This is not very fair.

However it would be even better if we can measure fairness with a number. You can do this with the *Gini* marker.

1.2 Gini index

To obtain the *Gini* from this flight architecture you can use the gini method implemented in kafkanator API. You can read comments to understand how to proceed.

```
from kafkanator import lorentz_curve,gini
import matplotlib.pyplot as pyplt
import numpy as np
# In order to use gini, you must pass one array with the income ordered from the
↪  people having less
# income to the high income population. In our case we have W*POP_3 people with
↪  confort 1, W*POP_2 confort 2 and W*POP_1 confort 1

conf1 = np.concatenate( ( np.repeat(CONFORT_3,W*POP_3) , np.repeat(CONFORT_2,W*POP_2)
↪  , np.repeat(CONFORT_1,W*POP_1)) , axis=None)
g = gini(conf1)
print (g)
0.24242424242424243
```

As a reminder, the closer the Gini index is to 0, the more equity there is in ressource assignation. You are invited to see in the notebook, how an increment of class 3 population to 360 people and 0.5 confort points instead of 1 impact Gini Index.

2 The salary distribution example

Let's imagine that the fictional corporation fairnessforworkers.com , hired you as a data scientist, in order to measure how fair they are doing when talking about salary distribution among workers. Specifically, they want to analize salary gap among salaries having the same speciality. They will give you the following dataset (that you can find in data / salaries.csv):

Identifier	Experience	Title (Diploma)	Salary
1	2	A	1000
2	3	A	2000
3	4	B	4000
4	5	B	3500
5	4	A	8000
6	1	B	2000
7	3	C	4000
8	4	C	5000
9	5	B	2000
10	1	A	5000

You can solve this problem, by mixing two statistical techniques : clustering and Gini index !. You can find all the code in tutorials / Salary Inequality.ipynb.

Lets use kafkanator API to quickly solve this type of problems, At the end you will get an array of Ginis, each one corresponds to the clusters generated by diplomas A,B,C:

```
from kafkanator import gini_per_cluster

workers = pd.read_csv('./data/salaries.csv',sep=',',header=0)

salary_groups = workers.groupby ( ['diploma']   )

gns = gini_per_cluster(workers,'diploma','salary')

print(gns)
gns = [('C', 0.05555555555555555), ('A', 0.375), ('B', 0.16304347826086957)]
```

From this result we can conclude that the most fair salary distribution is on the people belonging to diploma C cluster, while diploma A cluster is showing strong inequality in their salary distribution.

Note that this is a first approximation of the salary inequality problem. There are other important features to take into account when assessing fairess on this case. For example there could be people that according to his contract has to work more hours, so they are entitled to earn more money. In the next kafkanator versions I will refine the solution for these cases.

3 Example 3. Fairness in recommendation systems

Let 's imagine that you are a very cool and full of swag rock singer, and you want to broadcast your music videos on the internet. You want to choose a web video broadcast company that ensure you that you will appear a fair-equal number of times in the recommendations that they made to their customers compared to the other rock stars. You have choosen cooltube.com because it has a good reputation to be fair in the sense you are looking for.

However, is cooltube.com showing you a fair number of times in their recommendation system ? Which could be a fair number of appearances in their recommender system ? How we compute such a number ?.

We can model the coolsongs.com recommendation engine output as a table. One column for the query identification number, and another column for a list of artists ids that match that query by order of relevance from left to right .

Table 2 shows a table example , you can find the whole code in *tutorials / Fairness Matrix.ipynb*.

Query	Rankings
1	2,5,7,10,12
2	1,6,7,5,8
3	5,7,9,10,12
4	14,5,10,2,1
5	2,8,9,10,1
6	4,6,7,9,10
7	13,5,9,15,16
8	17,5,6,18,1
9	1,2,3,4,5
10	20,19,18,17,14

This table shows the output for 10 different queries. Row one means that the artist with id 2 was the most relevant, and it was shown first, then artist with id 5 and so on.

3.1 Recommendation Score for Artists

From such a table we can define the artist k recommended score (RA_k) by:

$$RA_k = \sum_{i=1}^{n}(N - pos_i(Artist_k))$$

Where N is the number of artist shown by output (5 in this case), n is the number of rows, and $pos_i(Artist_k)$, is the position on row i that artist k is holding from left to right , being 0 the position of the leftmost identifier. For example in query 5, $pos_5(Artist_9) = 2$.

RA_k can be used to verify how fair is being coolsongs.com recommendation engine.It measures how much an artist was shown in the recommendation system in a temporal sequence of queries.

Check the python notebook in order to see how **Algorithm 2** computes this score for every artist and leaves the result in rock_artist dictionnary. Once these scores are computed, you can apply unequality measures over these scores , like Gini index, Theil index etc, to analyze inequality.

3.2 Recommendation Score for Sensitive Attributes

Business also normally have a table with sensitive attributes of artists to detect bias in the recommendations :

artist id	age	gender	nationality
1	20	M	national
2	52	F	foreign
3	36	M	national
4	25	F	foreign
5	67	M	national
6	45	F	foreign
7	59	M	national
8	23	F	foreign
9	18	M	national
10	56	F	foreign
11	64	M	national
12	35	F	foreign
13	38	M	national
14	63	F	foreign
15	41	M	national
16	72	F	national
17	20	M	national
18	19	F	national
19	29	M	national
20	46	F	national

At the same pace we compute RAs, we can compute the sensitive attribute score. Intuitively, if an artist is man and foreigner and it obtained a score of

$$RA_k = \sum_{i=1}^{n}(N - pos_i(Artist_k)) = 23$$

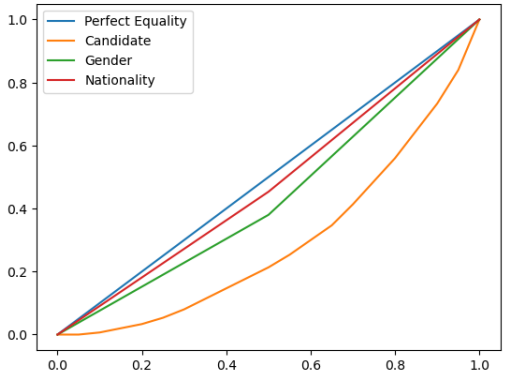
This same score can be added to the sensitive categories *man* and *foreigner*.

Formally, for sensitive values $\{SV_1, SV_2, SV_3, ...SV_n\}$ belonging to Sensitive Category SC_m (for example $\{Man, Womam\} \in Gender$), we can define the Sensitive value score SV_j like :

$$SV_j = \sum_{i=1}^{n}(N - pos_i(Artist_k) \textbf{ when } Artist_k[SC_m] == SV_j)$$

Where $Artist_k[SC_m]$ is the value of the column SC_m in artists table.

Inspecting the notebook, you can see how **Algorithm 2** computes also sensitivity weights and leaves them on counters_sens_attributes variable. Once this is computed, the notebook plots lorentz curves to study unfairness related to sensitive attributes :



From these curves, we conclude that Gender attribute deserves attention. as is the one that is far from the blue equity line.
(Revisar bien esta conclusion, 'nationality': 'foreign': 57, 'national': 93, 'gender': 'F': 68, 'M': 82)

References

[1] https://en.wikipedia.org/wiki/Lorenz_curve