# Definition of rough estimation of positivity rate

The goal is to use a fast, easy and versatile method to estimate the positivity rate of new customers.

## Customer inputs:

The customer should provide age, sex distribution and some other important factors distributions depends on the AlgoMarker - for example smoking status distribution for LungFlag. Or we can take it from national wide information of the region.

## Output

* Cutoff in several positivity rates (compared to our reference)
* Vice versa – positivity rate in out cutoffs
* (To be discussed if to share with client) – performance estimation: sensitivity , OR, PPV.   
  I assume that most customers that reach this point already trust us/our model is some way that we can deliver, and it’s less makes a different if the odds ratio is 9 or 7. There are already some negotiations with the customer, please provide us some distribution number, etc. It’s not a new vanilla client. If they don’t believe it will work on their data, why would they believe us after rough estimation? It can just make them get confused or encourage them to test us (which is more work for us, they might think of validation/why not?). I think that in this point their main concern is operational – don’t blow my queues and take all my resources, nag my physicians/patients with additional screening procedures. I’m ready to try, even if your AlgoMarker is not that better than “random” we are within guidelines. I am afraid from an operational perspective. Health care systems are good at guidelines, we need to vaccinate with covid all ages older than X. Let’s do it and measure compliance and operational sides of those vaccination campaigns (X% were vaccinated in target age range). It’s much rarer to measure impact and most organization don’t do it.  
  I might be wrong and I just wrote down those concerns, but I’m not strongly against presenting performance metrices in the output.

## Method

When there is 1 distribution it is easy – we can “match” the reference population by the distribution parameters to the target and “measure” in bootstrap the metrices we want – positivity rate/performance evaluations.

If we have more than one distribution, we need to match all of them.

2 Methods:

1. We can match them 1 after another – each matching might “break” the other one we did before. Advantage – simpler, independent.
2. We can do the matching “recursively” in each bin we want to “throw” aways random samples since they are over presented, we will stratify by the next distribution variable and throw from relevant bin/s to balance the other distribution (if the other distribution is already balances, we will throw away randomly). More complicated, creates some “biases” that depends on the order we iterate over the bins. Not sure we will end up balanced by all distributions but at least the first one will be respected, and it supposed to be better than 1 in respecting the other distributions.

## Deeper details

API:

Class StrataStat {

public:

/// Name of the feature like “Age”, “Gender” that match the strata we are going to match

string feature\_name;

/// Minimal value of the bin range

float min\_range;

/// Maximal value of the bin range

float max\_range;

/// The probability in the target to have this bin

float prob;

};

get\_weighted\_samples(const MedSamples &samples, MedRepository &rep, const string &json\_model, const vector<StrataStat> &stratas)

This function will receive Samples in original training repository and the training repository a json file to generate matrix from the samples. The matrix will be simple features like Age,Sex,Smoking status that we will be able to match “client” information.

The “stratas” is the descriptive statistics we received from the client and we need to match to => calculate the weights to match this distribution.

The output will store the weights in “sample”.

We can then use those samples with “scores”, “outcome” to estimate some bootstrap measurements like “cutoff” or performance.

Just running bootstrap over those samples.

Option to run on external => different function since we don’t want Repository:

get\_weighted\_samples(const MedFeatures &features, const vector<StrataStat> &stratas)

==== Suggestion:

Calculate also incidence rate base => allow to multiply by factor to adjust ppv results