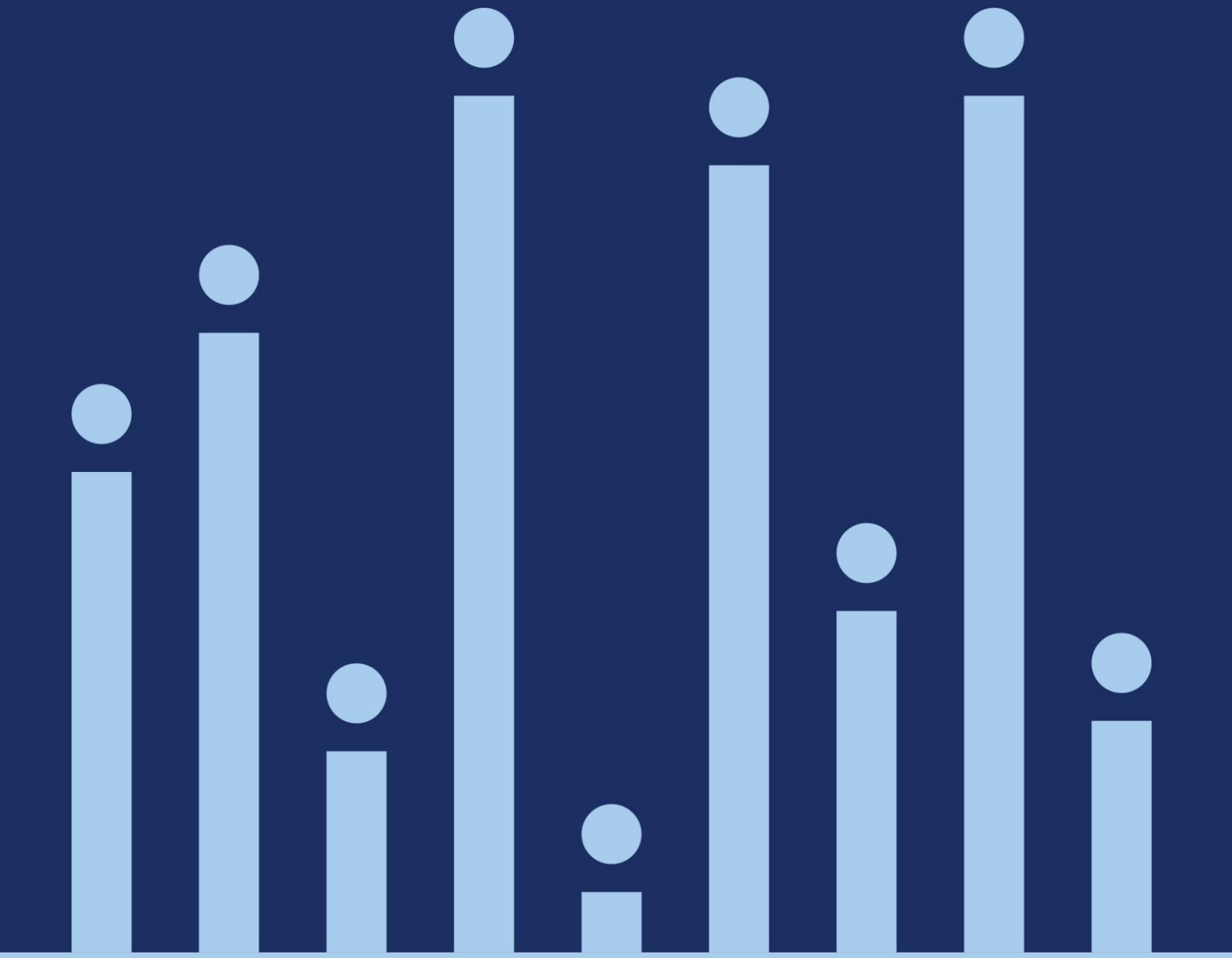


Matching eligible people to social welfare programs in India

September 2021

IDinsight

Contents

1 Background

India has many government welfare programs designed to transfer money to some of its most vulnerable people. But for a **number of these schemes**, people either don't know they are eligible, or don't apply and many of the funds allocated remain unspent. For example, only 40% of INR 50,000 crores has been spent to date in a fund dedicated to the welfare of construction workers. The Building and Other Construction Workers (BoCW) Act, implemented in 1996, provides a range of transfers including health, education, maternal, and pension benefits, for workers in this industry. However, less than 50% of the estimated construction workers in the country are registered with labour boards, and a smaller fraction has accessed these benefits to date.

The registration and claims processes for social benefits can be complicated and onerous, which is one reason why few eligible people access these benefits. A second reason is that the government does not know which citizens are eligible, and in turn, individuals don't know about all the schemes and their eligibility.

A number **of other groups** are working to improve people's access to services by creating platforms that help them determine their eligibility for different social programs. However, even if people are aware of the service, they may not have enough digital access or literacy to effectively use them. The people who are most vulnerable are least likely to get online to check their eligibility or have funds to pay for a service.

Indus Action, an India-based non-profit that supports households to gain access to legislated benefits, overcomes these challenges by reaching out to this population directly. Their outreach approach includes phone calls, SMS messages, and in-person visits. Through their direct work with individuals and collaborations with state governments, they have rich databases with information about individuals who may be eligible for different services.

Given Indus Action's available information about an individual, could we identify the set of benefits they would be eligible for? If we were able to build a system that answers this question of who might be eligible for what, we could then develop a smart outreach

campaign that could channel resources to where they would be most likely to reach eligible households.

Indus Action and our IDinsight team recently built this outreach system that, given a set of citizens and their characteristics, can predict the probability that a citizen is eligible for a benefit. A major feature of this system is that it can still make predictions about an individual's eligibility even if their data in the system is incomplete. The first set of benefits targeted through this system were those offered by the Building and other Construction Workers Act (BoCW) in Delhi.

In the rest of this report, we describe (1) the major components of the outreach system, (2) deploying the system as a serverless application on Amazon Web Services (AWS), and (3) potential to reuse this system for other benefits.

2 How the system works

We developed an outreach system that uses a roster of *benefits* and their requirements and the attributes of a list of *individuals*. The system then returns eligibility scores for all the (*benefits* x *individuals*) combinations.

Often, multiple eligibility criteria need to be met by an individual to qualify for a benefit. For example, to qualify for the educational benefit under BoCW, a construction worker should have been registered with the labour board for at least one year and should have proof of admission of their children to a school. However, the government doesn't know whether a worker has children, as they did not collect this detail at the time of registration. How can the government now use available information to help the workers access the benefit without asking them to provide more data than they have already provided?

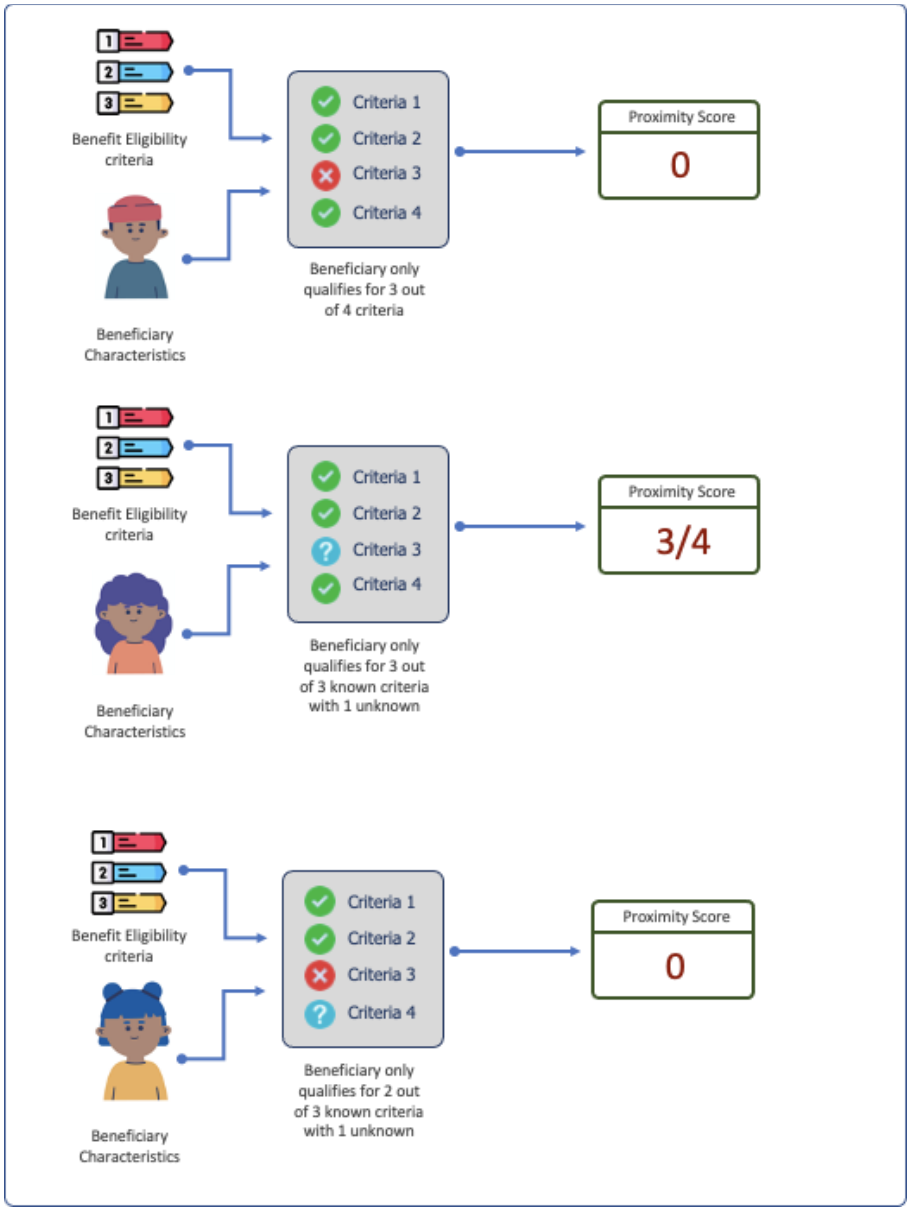
While matching individuals with benefits with incomplete information, we considered two kinds of reasoning. First, given our existing knowledge of each individual, we were able to determine who is eligible/ineligible for some benefit whenever the criteria for that benefit matched known information about the individual. Second, if the information needed to determine eligibility was missing along some axes of benefit criteria, then there was uncertainty whether an individual was eligible for criteria. In such cases, it may be possible to infer this individual's eligibility for the benefit using other information we know about this individual.

However, for decision-makers to prioritise which individuals to call, we needed to balance certainty and uncertainty. To help facilitate this, the outreach system computes two scores.

The first score that the system produces is the **Proximity Score.** This score indicates the certainty of a person's eligibility. If the value is -1 an individual does not qualify for a benefit; if the value is 1 the individual does qualify given existing information. Furthermore, it indicates how close an individual is to fully qualifying for a benefit. For

each benefit, if we do not know whether the individual qualifies or not, the score is the percentage of benefit criteria an individual is known to fulfill.

Figure 1: *Calculating the proximity score*



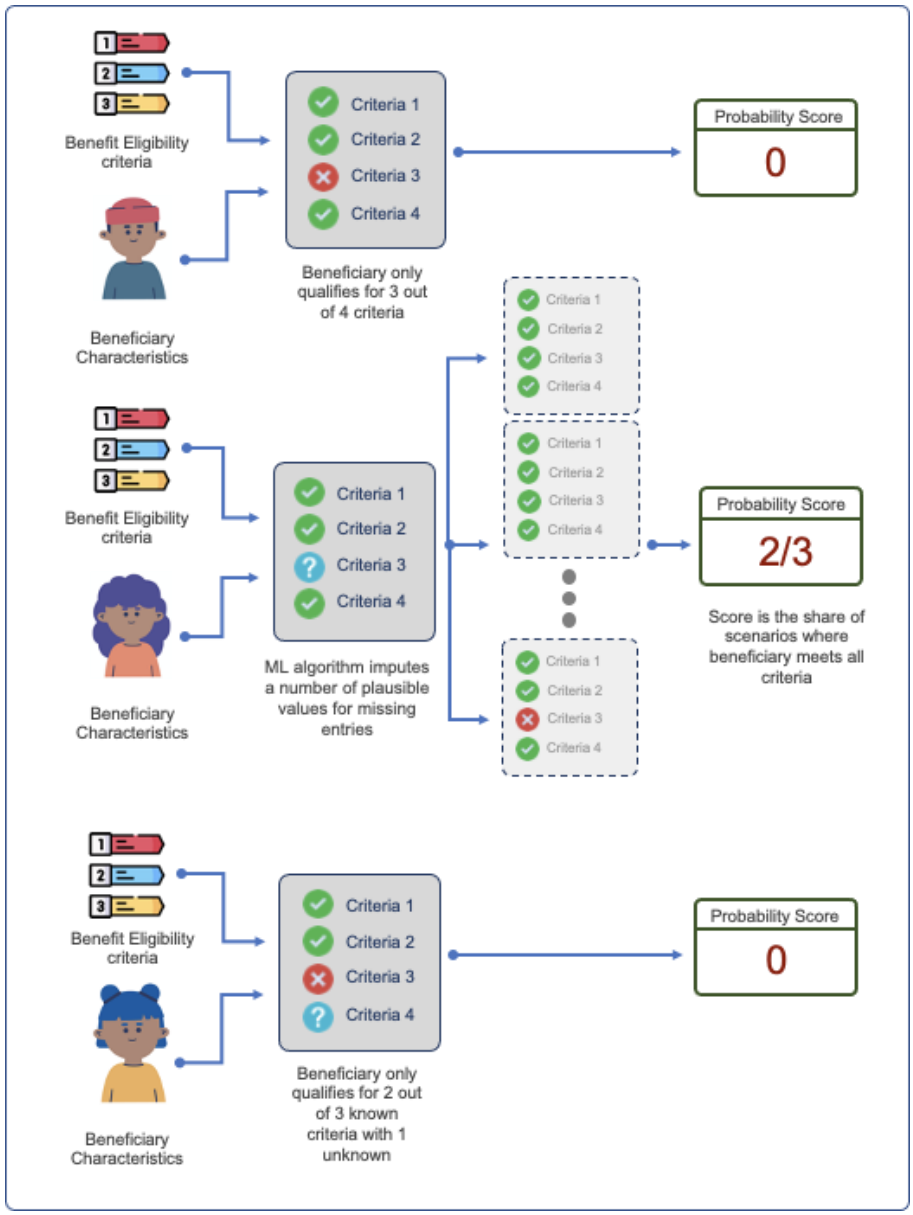
The second score that the system produces is the **Probability Score**. It represents the likelihood of an individual qualifying for a benefit given (a) the incomplete set of characteristics of the individual; and (b) the distribution of characteristics of other individuals. The system employs matrix completion techniques to impute the missing individual attributes.

Common matrix completion methods work best when the data missing is **completely at random**. We know this not to be true with our data since the data was collected in different campaigns which targeted specific benefits. For example, a **Right to Education** campaign might have only targeted parents while a **PM Kisan** campaign might have reached out to mostly farming communities. We combat this by capturing this uncertainty in the *Probability Score*. The system currently computes probability scoring

through both frequentist (bootstrapping the data to get a distribution of scores) and Bayesian techniques to generate posterior distributions.

There are two reasons to supplement the Proximity Score with the Probability score. First, if an individual does not meet the criteria for a benefit, the unmet criteria are weighted equally. Weighing all criteria equally does not allow us to account for cases where some criteria are likely easier to meet than others. For example, an individual may be able to get a bank account but is likely to still not meet other criteria such as age or occupation. The system should have the ability to weigh these criteria differently. Similarly, for criteria that we do not have data on, we know that an individual is likely to meet some criteria more than others. For example, say there is a 35-year-old individual, but we don't know if they are (a) married or (b) are a construction worker. The probability that the individual is married is likely higher than that of them being a construction worker. The ability to reason probabilistically under uncertainty given limited information can be valuable when prioritising who to contact for an outreach campaign.

Figure 2: *Calculating the probability score*



Neither of these scores are perfect. But when used together, they can inform the design of outreach campaigns. Decision-makers can prioritise individuals to contact by balancing known eligibility with uncertainty.

3 Deploying the Outreach System

As a major part of the project was to take the outreach system from a proof-of-concept (PoC) to a live production system, we needed to understand how Indus Action wanted to use the outreach system to support its programmes. When building any sort of software, it is important to put the end-user first and build around their needs. A standard method to build an understanding of the user is through user stories, which are descriptions of how a solution or feature would drive value for the user. Through co-design sessions, we worked with the Indus Action team to develop three user stories.

These user stories were:

1. *Inbound live messaging:* User messages an Indus Action chatbot which calls our API to create a custom message letting them know which benefits they are eligible for.
2. *Outbound bulk automated messages via WhatsApp/text messages:* Indus Action generates *proximity* and *probability* scores for a number of selected individuals for contact. Those scores determine which individual gets what message as part of a bulk message campaign.
3. *Scheduled jobs to score potential beneficiaries:* Score a large number of individuals and store their data for Indus Action's internal outreach teams to prioritise

To create a proof of concept for these user stories, we designed a serverless solution in AWS for the outreach system. This decision came about after discussions with Indus Action around key design principles around minimising cost, fast response times, and the ability to scale up to include more benefits and individuals.

3.1 Serverless Design

Serverless design on the cloud refers to an architecture based around managed resources that are allocated on demand. That is, resources which only charge money during computation. Often serverless design is supported by "code as infrastructure" and set up to be scalable with an emphasis on parallel orchestration.

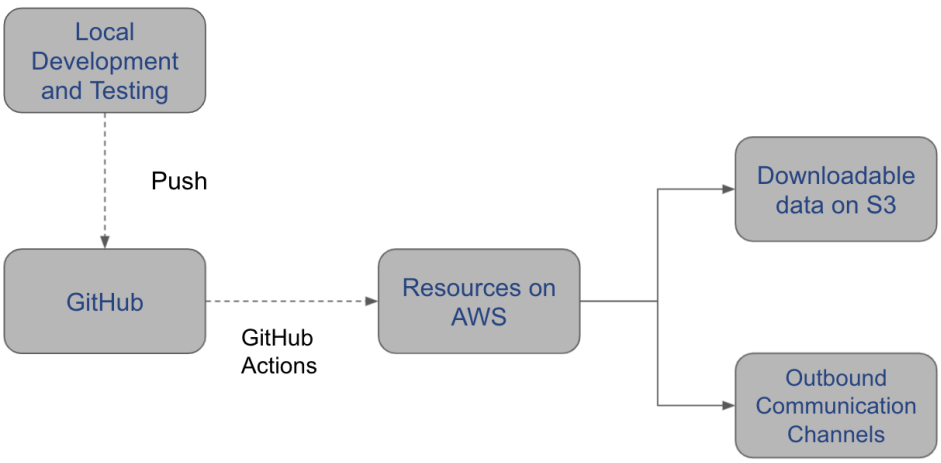
Table 1: advantages and disadvantages of a serverless design on the cloud

Advantages	Disadvantages
No need to manage servers.	Constraints imposed by vendor-managed resources. For example, capped memory limits and timeouts.
Easy deployment. Usually completely controlled in a few config scripts.	Dependence on a particular cloud vendor.
Cheaper due to demand-based costing and easier to scale due to serverless resources usually designed to run in parallel or async.	Serverless resources are often less performative due to additional overhead.

3.2 Infrastructure

We used AWS Serverless Application Templates (SAM) with GitHub actions to deploy our solution to the cloud. This enabled us to spin up CI/CD (continuous integration / continuous development) with minimal effort as building the PoC took place in parallel to the creation of other features for the solution.

Figure 3: Development workflow



Changes in local deployment, when pushed to the cloud, automatically refresh deployed resources in AWS to reflect those changes. This enabled large scale feature changes such as the inclusion of bootstrap imputation to take place over a few days as opposed to a couple of weeks.

3.3 Solution Architecture

Our solution on AWS uses just three resources. We use *AWS Lambda* for computation orchestrated by *AWS Step Functions* with flat files stored in *S3*. This minimal architecture supports the scoring of hundreds of thousands of individuals.

We will go through the architecture whilst linking it back to the user stories.

3.3.1 Live Scoring

For live scoring, we assume that on the Indus Action side, the chatbot will be calling an endpoint to fetch individual scores and post individual information. As such, we simply create an *AWS API Gateway* endpoint attached to a Lambda which handles such requests and returns scores. This enables user story 1.

3.3.2 Data Processing and Probability Scoring

Creating probability scores via bootstrap is an expensive operation and quite time-consuming. As such, we should only do it when truly required. Since probability scores are not likely to meaningfully change unless the data is updated, we run probability scoring as part of the data processing pipeline. This pipeline fetches data from S3, converts it into a scorable form, and probability scores are calculated. This pipeline is run only when the data in S3 is updated.

3.3.3 Bulk Scoring

To match individuals with benefits, Indus Action would need the ability to bulk score a large amount of data. We use step functions to parallelise individual proximity scoring by asynchronously executing Lambdas by benefit. This is possible because, at this stage of the processing, eligibility for each benefit does not affect eligibility for other benefits. Scored individuals are then written back into S3 for Indus Action to use for outreach activities. This enables user story 3. When extended by adding the outbound capabilities of *Live Scoring,* this also enables user story 2.

In the step functions above, we make full advantage of the parallel execution capabilities of Lambda. This allowed us to make significant performance improvements in time (~50x improvement when moving from local to the cloud).

4 Potential to reuse the system for other benefits

The information barriers the solution was designed to overcome are not specific to the BoCW Act in Delhi. These barriers contribute to the low take-up of many schemes in many states. Maternity benefits under PMMVY are **one** example. Fortunately, the solution is generic and configurable enough that it can be extended to other such benefits easily.

While we believe that this (solution) will make a huge impact on its own, steps to simplify the application and related processes could also make a big difference. Specifically, for BoCW, Indus Action and state governments are diagnosing documentation-related challenges for migrant workers, setting up user-friendly interfaces for application, and strengthening processes for doorstep delivery to increase registration of construction workers.

If you are interested in learning more about this project or how machine learning and engineering can help you improve your operations, please reach out to the Data Science, Engineering, and Monitoring Systems team at IDinsight at ***dsem@idinsight.org****.*

www.IDinsight.org
@IDinsight

IDinsight