

# rarelens

An end-to-end rare-disease variant triage platform

A technical introduction — pipeline, infrastructure, model and the evidence behind the ranking

---

## In one paragraph

**rarelens takes a patient's genome and their clinical phenotype and narrows thousands of variants to a handful a scientist can actually review, showing the evidence behind every rank.** It is a working monorepo: a Nextflow DSL2 pipeline that annotates variants with Ensembl VEP, a FastAPI service that ranks them against Human Phenotype Ontology annotations, a LightGBM model served from an MLflow registry, a SvelteKit interface built around triage decisions rather than table filtering, and Terraform for two deployment tracks on Google Cloud — a serverless one that idles at roughly £1 a month, and a Kubernetes one with Argo Workflows, Argo Events and ArgoCD behind a feature flag. Everything runs on public, openly licensed data. No patient data is used, accepted, or possible to load.

This document is a technical tour. It is also, deliberately, an account of the things that turned out to be wrong — including a model metric that collapsed from 0.872 to 0.500 the moment I took away the feature that was doing all the work, and an ontology bug that silently deleted 399 terms.

## 1 The problem

A rare-disease proband's exome contains something like twenty thousand coding variants. Perhaps a few hundred are rare and protein-altering. Exactly one, usually, explains why the patient is ill. Finding it is not a filtering problem so much as an evidence problem: the same variant is uninteresting in one patient and diagnostic in another, and what changes between them is the clinical phenotype.

That observation drives the whole design. The interface is not a variant table with filters — it is a ranked shortlist where each candidate carries the four pieces of evidence that put it there, and where a reviewer shortlists or dismisses with a reason that ends up in a case report.

## 2 The Nextflow pipeline

Annotation is a three-process DSL2 workflow, each process a pinned container: `bcftools` normalises, Ensembl VEP 113 annotates, and a loader writes results into PostgreSQL and marks the job succeeded.

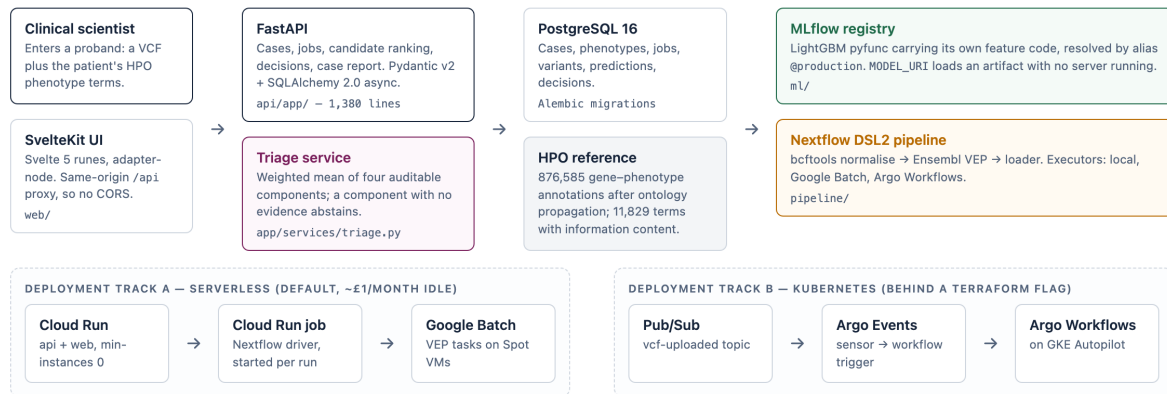
Two details are worth pulling out, because both cost real debugging time.

### 2.1 Variant identity survives annotation

VEP's `Location` and `Allele` output columns trim indel alleles and shift positions by one, which is correct for VEP's own purposes and wrong as a primary key. A deletion written by the caller as `CT>C` comes back as `->` at a different coordinate, and the loader then stores a variant that does not exist in the input file.

## rarelens — system architecture

One monorepo: scientific pipeline, API, UI, model serving and two deployment tracks. Arrows are data flow.

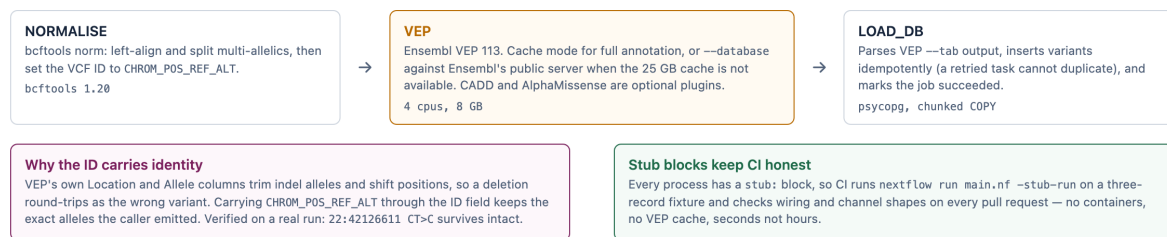


Both tracks run the identical pipeline code; the executor is a Nextflow profile, and ArgoCD reconciles the Kubernetes track from the same repository.

**Figure 1:** The system. One repository holds the pipeline, the API, the interface, model serving and the infrastructure for two deployment tracks. The scientist never sees any of it except the middle column.

## The Nextflow DSL2 pipeline

Three processes, each a container. The same workflow file runs on a laptop, on Google Batch and under Argo Workflows — the executor is a profile, not a rewrite.



**Figure 2:** The pipeline. The same workflow file runs on a laptop, on Google Batch and under Argo Workflows; the executor is a profile, not a rewrite.

The fix is to carry identity through a field VEP does not touch. **NORMALISE** sets the VCF ID to CHROM\_POS\_REF\_ALT, and the loader parses identity back out of it:

```
bcftools norm -m -any -f $genome | \
  bcftools annotate --set-id '%CHROM\__%POS\__%REF\__%FIRST\_ALT'
```

Verified on a real run: 22:42126611 CT>C round-trips with its alleles intact.

## 2.2 Stub blocks make continuous integration possible at all

A VEP cache is 25 GB. No pull request is going to download one. Every process therefore carries a **stub:** block, and CI runs the workflow with **--stub-run** against a three-record fixture, checking channel wiring and process contracts in seconds without a container or a cache in sight. It catches the failure that actually happens in practice — a renamed output, a channel that emits the wrong cardinality — while leaving the scientific correctness to the tests that can afford to be slow.

### 3 Event-driven execution, three ways

`POST /cases/{id}/annotate` writes a job row and hands off. What happens next is a configuration decision, not a code path the caller knows about.

#### One API call, three execution backends

`POST /cases/{id}/annotate` creates a job row, then `events.launch()` picks a backend from configuration alone. The pipeline code never changes.



The database URL never appears on a command line: it is passed by environment, or as a Nextflow secret, so it stays out of `.command.sh` and the workflow logs.

Figure 3: One entry point, three backends, chosen from settings alone.

- **Cloud Run job.** The Nextflow driver runs as a Cloud Run job started through the Jobs API with argument overrides. It scales to zero between runs, and its service account holds `run.jobsExecutorWithOverrides` on exactly one job — not project-wide.
- **Pub/Sub.** The job is published as an event. An Argo Events sensor subscribes and triggers an Argo Workflow on GKE. This is the decoupled path: retries, ordering and back-pressure become the queue's problem rather than the API's, and the API can be restarted mid-pipeline without losing work.
- **Local subprocess.** Nextflow runs directly and its stdout is streamed into the job log, so the interface shows live progress. This is what a developer gets with nothing configured, and it is the same code path the other two wrap.

All three converge on the same `jobs` row, so the interface polls one endpoint regardless. The database URL is passed by environment or as a Nextflow secret and never appears on a command line, keeping it out of `.command.sh` and the workflow logs.

### 4 Kubernetes, Argo and GitOps

The Kubernetes track is Kustomize bases with a local overlay (kind, an in-cluster Postgres) and a GCP overlay (Cloud SQL, Workload Identity). Argo Workflows runs the annotation `WorkflowTemplate`; Argo Events holds the Pub/Sub EventSource and the sensor that triggers it; ArgoCD reconciles the cluster from the repository, and a green CI run on `main` bumps image tags in the GCP overlay so that deployment is a commit rather than a command.

One bug from this area is worth recording because it is invisible until it bites: Kustomize generates hashed ConfigMap names so that a configuration change forces a rollout, but the hashed name is only substituted into workloads Kustomize believes are in scope. The overlays were missing `namespace: rarelens`, so the substitution silently did not happen and pods mounted a ConfigMap name that no longer existed. The symptom was a pod stuck in `CreateContainerConfigError` with nothing wrong in the manifests as written.

## 5 Infrastructure as code

Terraform provisions both tracks from one root module, with the expensive half behind flags:

```
terraform apply -var project=<id>                                # serverless: Cloud Run
+ Batch
terraform apply -var project=<id> -var deploy_kubernetes=true \
                -var deploy_cloud_sql=true                      # adds GKE, Argo, Cloud
                                                                SQL
```

The default track provisions Cloud Run services for the API and interface, a Cloud Run job for the Nextflow driver, Google Batch for pipeline tasks on Spot VMs, a GCS bucket, Secret Manager entries and Artifact Registry. GKE Autopilot and Cloud SQL are opt-in, because a Kubernetes control plane and a managed database are most of what a demonstration estate costs.

CI authenticates to Google Cloud through Workload Identity Federation, so there is no service account key anywhere in the repository or in GitHub secrets. Terraform is validated and format-checked on every pull request.

### 5.1 Cost as a design constraint

A portfolio platform is idle more than 99% of the time, which makes idle cost the only cost that matters. The serverless track is built around that: Cloud Run at `min-instances=0`, a driver that exists only while a pipeline runs, and Batch on Spot. Idle cost lands near £1 a month, almost all of it the database. The Kubernetes track exists to demonstrate the GitOps path and is meant to be destroyed afterwards.

## 6 Integrating external systems

Almost nothing here is self-contained, and integrating public biological infrastructure is most of the work:

- **Ensembl VEP** in cache mode, or against Ensembl's *public database server* with `-database` when 25 GB is not available — slower per variant, no plugin scores, but no download.
- **gnomAD v4.1** allele frequencies streamed by genomic region straight out of the public Google Cloud bucket. The files are tabix-indexed, so a range request returns a few hundred records without fetching the file. I had previously written off frequencies as impossible without the full cache; testing that assumption disproved it.
- **Human Phenotype Ontology** gene-to-phenotype annotations plus the ontology itself, propagated and weighted at load time.
- **ClinVar** for model training labels, filtered to two-star review status and above.
- **Phenopacket Store** (Monarch Initiative), which curates published case reports into GA4GH phenopackets — the source of both the demonstration case and the benchmark.
- **MLflow** as a model registry, resolved by alias, with a registry-free path for deployments that should not run a tracking server.

```
# real gnomAD frequencies with no bulk download, verified end to end
bcftools view -r chr3:30672000-30673000 \
```

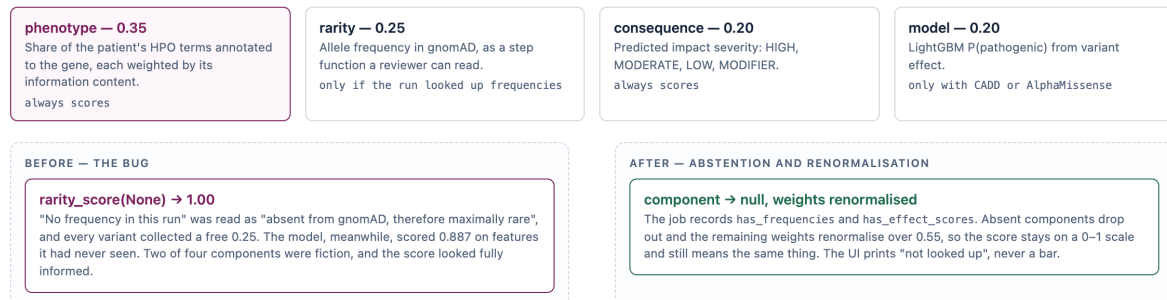
```
https://storage.googleapis.com/gcp-public-data--gnomad/release/4.1/...chr3.vcf.
bgz
# -> 392 records, 5 KB, then fed to VEP with --custom alongside --database
```

## 7 The ranking, and what it refuses to claim

The rank is a weighted mean of four components a reviewer can audit. ClinVar is deliberately *not* one of them — it sits beside the result as independent confirmation, so nothing ranks highly merely because ClinVar already called it pathogenic.

### Evidence that was never looked up must abstain

The rank is a weighted mean of four auditable components. Which ones may score is decided per job, from what the annotation run actually produced.



| Published Loëys–Dietz case, run without a VEP cache      | score | phenotype (0.64) | rarity        | consequence (0.36) | model         |
|----------------------------------------------------------|-------|------------------|---------------|--------------------|---------------|
| TGFBR2 3:30672252 G>T missense — the published diagnosis | 0.855 | 1.00 (30/30)     | not looked up | 0.60               | not looked up |
| OSBPL10 3:31748090 missense — incidental                 | 0.218 | 0.00             | not looked up | 0.60               | not looked up |

Both are rare missense variants, identical on every piece of evidence this run holds except one. The phenotype is what separates a published diagnosis from an incidental variant in a lipid-transport gene — which is the argument for phenotype-driven triage, in one table.

**Figure 4:** Rarity and consequence filter; phenotype only ranks, because a real diagnosis can sit in a gene nobody has annotated yet. A component with no evidence behind it abstains, and the remaining weights renormalise.

The abstention rule replaced a genuine bug. Run without a VEP cache, there are no allele frequencies — and the code read a missing frequency as *absent from gnomAD, therefore maximally rare*, handing every variant a free quarter of its score. The model, separately, was returning 0.887 for every variant on features it had never been given. Two of four components were fiction, and the total looked fully informed.

The fix is structural rather than cosmetic: the loader records what the annotation run actually produced (`has_frequencies`, `has_effect_scores`), components without evidence return null instead of a number, and the weights renormalise over whatever is left. The interface prints “not looked up” where it would otherwise have drawn a bar.

## 8 Measuring the ranking

One demonstration case ranking correctly is an anecdote. The benchmark asks the only question that matters for a phenotype-driven tool, across every usable case in Phenopacket Store.

Two things in that figure matter more than the headline. The first is the contamination: HPO’s gene annotations are curated from these same case reports, so the median causal gene already carries every one of its patient’s terms. The number is an upper bound and is labelled as one. The second is that when I measured my own improvements, one of them did not work — information-

### Does the phenotype ranking actually work?

Every case in Monarch's Phenopacket Store: given a real patient's reported terms, where does the gene their authors diagnosed rank among all 5,269 HPO-annotated genes? Ties give a range — optimistic counts a tie as a win, pessimistic counts every tied gene as ahead.

#### CAUSAL GENE RANKED FIRST (TOP-1), 10,178 PUBLISHED CASES



#### CAUSAL GENE IN THE TOP TEN



#### The benchmark is contaminated, and it must be said out loud

The median causal gene already carries every one of its patient's terms, because HPO's gene annotations are curated from these same case reports. This measures how well the ranking retrieves a gene HPO has already been told about: an upper bound. A prospective number, on a patient whose gene nobody has annotated yet, would be lower, and this corpus cannot say by how much.

#### Measuring my own changes, including the one that failed

Information-content weighting and ontology propagation both replaced plain term counting. Asked whether they helped, the corpus said only one of them did — pessimistic figures, 6,485 cases with six or more terms:

|                              |              |              |
|------------------------------|--------------|--------------|
| count terms (original)       | 61.8%        | 0.682        |
| <b>+ information content</b> | <b>63.6%</b> | <b>0.706</b> |
| + propagation                | 58.2%        | 0.653        |
| <b>+ both (shipped)</b>      | <b>59.5%</b> | <b>0.670</b> |

Weighting earns its place. Propagation costs about what weighting gains — kept for a reason the documentation argues rather than assumes, with the table there so a reader can disagree.

**Figure 5:** Retrieval against 5,269 candidate genes, and an honest report of what the measurement cannot tell you.

content weighting helped, ontology propagation cost about as much as weighting gained. That result is in the documentation with the table, rather than quietly dropped.

## 9 The model, and the number that changed it

The pathogenicity model is LightGBM trained on ClinVar labels, held out *by gene* rather than by variant. That distinction is not pedantry: a random split puts variants of the same gene on both sides, and the model then scores the gene instead of the variant, which is exactly the inflation Grimm *et al.* documented for this class of tool in 2015.

### The number that changed what the model is allowed to do

Held-out evaluation with whole genes held out, never single variants (Grimm *et al.* 2015): 312,025 training and 74,239 test variants across 7,728 and 1,932 genes, with no gene on both sides.

| MODEL                                          | AUROC | AUPRC | MISSENSE AUROC | MISSENSE AUPRC |
|------------------------------------------------|-------|-------|----------------|----------------|
| v2 — with gnomAD allele frequency as a feature | 0.986 | 0.954 | 0.872          | 0.725          |
| v3 — allele frequency removed                  | 0.966 | 0.881 | 0.500          | 0.398          |

**0.500**

AUROC on missense variants once frequency is removed. Exactly random.

#### What that proves

Strip frequency out and the model cannot tell one missense variant from another at all — nothing is left but the consequence class, so every missense row scores identically. The respectable-looking 0.872 was never variant-effect knowledge. It was allele frequency.

#### And the frequency feature was circular

ACMG's BA1/BS1 criteria assign ClinVar's *benign* labels using allele frequency. The feature had partly caused the label, so the model was rediscovering the rule that produced its own training data.

#### The consequence for the product

Frequency is no longer a feature — the ranking already scores it explicitly and audibly, and feeding it to the model as well put ~45% of every rank on one measurement counted twice. The model now abstains unless it has CADD or AlphaMissense, because 0.500 is the measurement saying it has nothing else to add.

**Figure 6:** Removing one feature moved missense AUROC from 0.872 to 0.500.

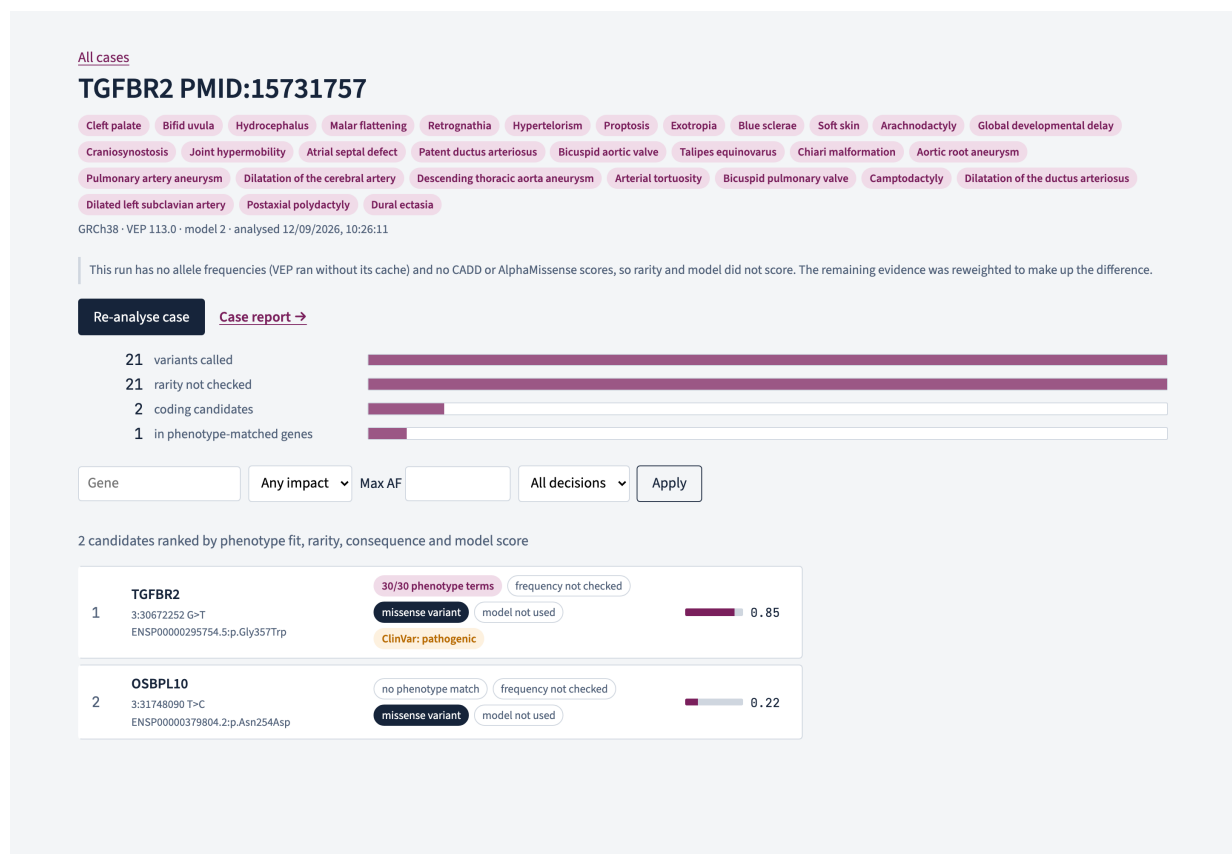
The model originally took allele frequency as a feature and looked respectable. Probing it showed frequency dominating everything — the same missense variant scored 0.887 at frequency zero and

0.0003 at one per cent. That is two separate problems. It double-counted, because the ranking already scores frequency explicitly, putting roughly 45% of every rank on one measurement. And it was circular, because ACMG’s BA1/BS1 criteria assign ClinVar’s benign labels *using* allele frequency, so the model was rediscovering the rule that had generated its own labels.

Retraining without it returned missense AUROC 0.500 — exactly random. With frequency gone and no CADD or AlphaMissense scores in the training table, nothing is left but the consequence class, so every missense variant scores identically. The conclusion is unambiguous and slightly uncomfortable: the model never had variant-effect knowledge. It now abstains from the ranking unless it has a predictor the other components do not already provide, and 0.500 is the measurement that justifies the abstention.

## 10 The application

The interface is built around the narrowing and the decision, not around the table.



**Figure 7:** A published case. The funnel across the top is the story — variants called, rarity, coding candidates, phenotype-matched — and each candidate carries its evidence as chips. The note under the header states plainly which components did not score and why.

## 11 Engineering practice

- **Tests where the risk is.** 99 Python tests plus 49 in the front end, concentrated on the ranking arithmetic, the ontology handling, the loader’s idempotency and the security boundary. Database tests run against a real PostgreSQL — an embedded server locally, a service container in CI — because the schema is part of the behaviour.

TGFB<sup>R</sup>2 3:30672252 G>T

30/30 phenotype terms

frequency not checked

missense variant

model not used

ClinVar: pathogenic

WHY IT RANKS 0.85

|             |                               |       |
|-------------|-------------------------------|-------|
| phenotype   | 1.00 × 0.64                   | 0.636 |
| rarity      | not looked up – did not score |       |
| consequence | 0.60 × 0.36                   | 0.218 |
| model       | not looked up – did not score |       |

PHENOTYPE MATCH

HPO annotates **TGFB<sup>R</sup>2** with Cleft palate, Bifid uvula, Hydrocephalus, Malar flattening, Retrognathia, Hypertelorism, Proptosis, Exotropia, Blue sclerae, Soft skin, Arachnodactyly, Global developmental delay, Craniosynostosis, Joint hypermobility, Atrial septal defect, Patent ductus arteriosus, Bicuspid aortic valve, Talipes equinovarus, Chiari malformation, Aortic root aneurysm, Pulmonary artery aneurysm, Dilatation of the cerebral artery, Descending thoracic aorta aneurysm, Arterial tortuosity, Bicuspid pulmonary valve, Camptodactyly, Dilatation of the ductus arteriosus, Dilated left subclavian artery, Postaxial polydactyly, Dural ectasia.

EVIDENCE

|             |                               |
|-------------|-------------------------------|
| Consequence | missense_variant MODERATE     |
| HGVS        | ENSP00000295754.5:p.Gly357Trp |
| gnomAD      | absent                        |
| ClinVar     | pathogenic                    |
| Model       | 0.89 (model 2)                |

[Ensembl](#) [gnomAD](#) [ClinVar](#)

DECISION

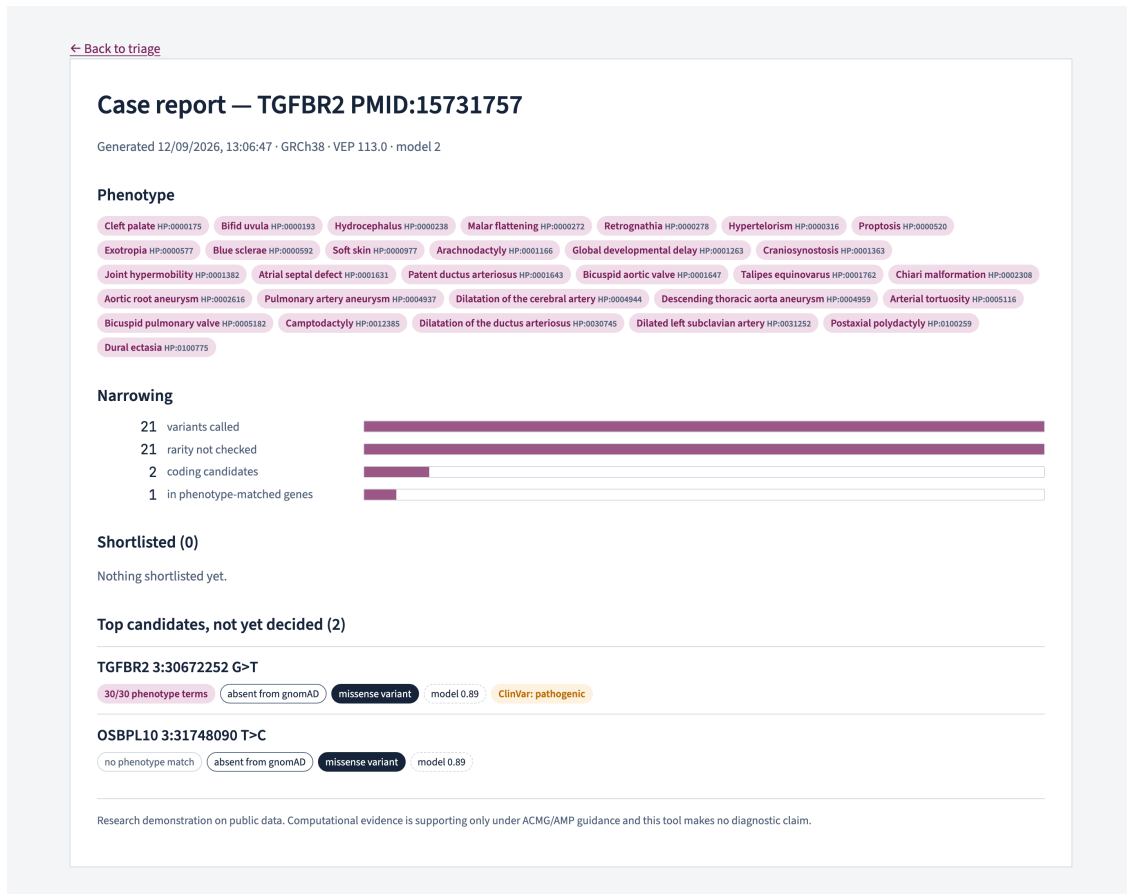
Reason, e.g. fits the phenotype

Note for the report

8

Shortlist

Dismiss



**Figure 9:** The case report: shortlisted and dismissed variants with the reviewer’s reasons, the funnel counts, and provenance — VEP version, model version, run time — so a result can be reproduced or challenged later.

- **Migrations are tested, both ways.** Alembic upgrade and downgrade are exercised against a live database, which is how a downgrade that left a stale enum type behind was caught.
- **CI runs the whole estate:** lint and types for the API, unit tests for the model code, a real Postgres for the loader, a Nextflow stub run, and `terraform validate`.
- **Input validation as a security boundary.** A case’s `vcf_uri` must be a `gs://` object or an absolute path beneath a configured data root, with a VCF suffix. That is what stops a crafted path becoming a Nextflow option or reading an arbitrary file, and it has its own test file.

## 12 Deployment

The demonstration runs on Railway — three services, with the API reachable only on the private network so the interface’s `/api` proxy is the single public entry point. One shared credential sits in front of it.

Railway cannot run the pipeline: Nextflow shells out to `docker run` for VEP and bcftools, and a container platform gives you a container, not a Docker daemon. Rather than leave a button that always fails, the cases are annotated where the pipeline works and copied up, and the pipeline actions are hidden. Making it idle correctly needed one real change — the platform decides a service is idle from its *outbound* traffic, and a pooled database connection is outbound traffic, so the default connection pool would have kept the service awake and billable for ever.

## 13 What is still wrong

An honest introduction should end with the open problems, not the achievements.

- **The model has no features in its training data.** Its only two remaining inputs, CADD and AlphaMissense, are absent from all 688,362 training rows. It abstains today, so nothing is broken — but installing the VEP plugins would flip it on while it still knows nothing. Serving should refuse unless the model was trained on the features it is handed.
- **The benchmark is contaminated** and no amount of code fixes it. A leave-one-publication-out rebuild is possible — HPO's annotation file carries the source PMID — and is the honest next step.
- **No baseline comparison.** Scoring the same held-out rows with CADD and AlphaMissense would give the model something to beat. AlphaMissense is a 0.64 GB download and tractable; CADD's whole-genome file is 87.5 GB and is not.
- **Rarity contributes nothing without the cache** in the default demonstration, though the gnomAD streaming route above now makes that solvable.

---

rarelens is a self-training project built in the open on public data, licensed AGPL-3.0. It is not a clinical tool and makes no diagnostic claim: ACMG/AMP treats computational predictions as supporting evidence only, never sufficient alone. Data sources, licences, citations and the evaluation caveats are documented in `docs/data.md`; the architecture decisions, including why Google Cloud rather than AWS, are in `docs/architecture.md` and `docs/cloud.md`.