
SNAP PER Analytics Workgroup

Implementing a predictive model within Connecticut

Anthony Lollo · 5.11.2026

Takeaways for Implementation

01

Training a model is $> \frac{1}{2}$ the battle

Training a model required linking data, understanding the data generating process, and setting up analytic environments.

Implementing the model reused much of that, just with **current data**.

02

The model output can be simple

Our model produces a scalar measure of risk for each case.

We can provide that output in an excel spreadsheet, or just as a list of cases to review.

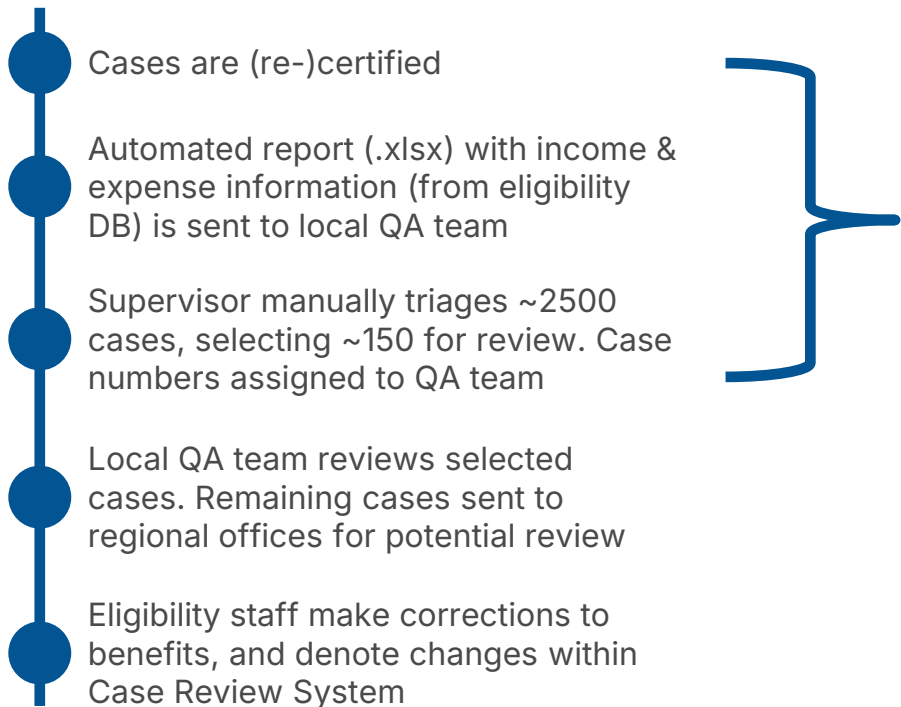
03

Put some energy into real-world evaluation

Implementing something that doesn't work isn't useful. Operational challenges may influence model impact.

Are there processes in place to fix the errors which are being predicted?

Starting Point – CT already triages cases for review

- 
- Cases are (re-)certified
 - Automated report (.xlsx) with income & expense information (from eligibility DB) is sent to local QA team
 - Supervisor manually triages ~2500 cases, selecting ~150 for review. Case numbers assigned to QA team
 - Local QA team reviews selected cases. Remaining cases sent to regional offices for potential review
 - Eligibility staff make corrections to benefits, and denote changes within Case Review System

1. Identify newly certified cases in eligibility systems and pull all relevant data for those cases into modeling environment
2. Run cases through model and generate predicted risk scores
3. Use scores to generate an .xlsx with a list of cases for review

*Do this in a way that preserves the current process, so we can do a **randomized, real-world comparison** to evaluate model use

Two Key Challenges: Automation and Workspaces

Incompatible Workspaces

Eligibility systems are SQL databases; predictive model was developed on a virtual machine using python.

Case review system is an entirely separate database

Long-term Goal

Integrate predictive model into eligibility and review systems

Automation

Many of the current steps involve manual intervention, especially with data transfers between teams and systems.

Data scientists aren't data engineers

Immediate Goal

Single "run" command to generate the formatted list of cases for a specified week

What implementation looks like in reality

1. Receive a .xlsx report with weekly cases (standardized naming/format)
2. Place that report in a project directory
3. Run a script to clean/standardize that report for use with model

```
import_raw.clean_weekly_list(date='05.01.2026')
```

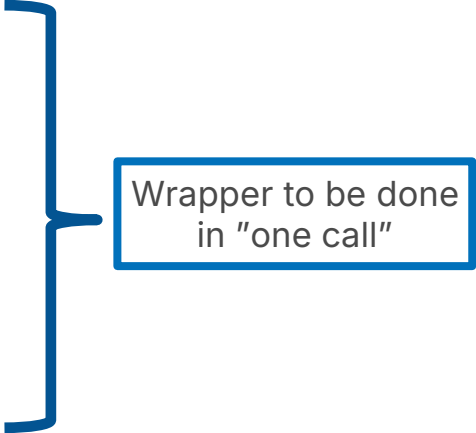
4. Use standardized report to query data from eligibility systems

```
query_eligibility.gen_weekly_features(date='05.01.2026')
```

5. Generate risk predictions and list of cases to review

```
predictive_model.gen_weekly_predictions(date='05.01.2026')  
predictive_model.gen_weekly_review_list(date='05.01.2026')
```

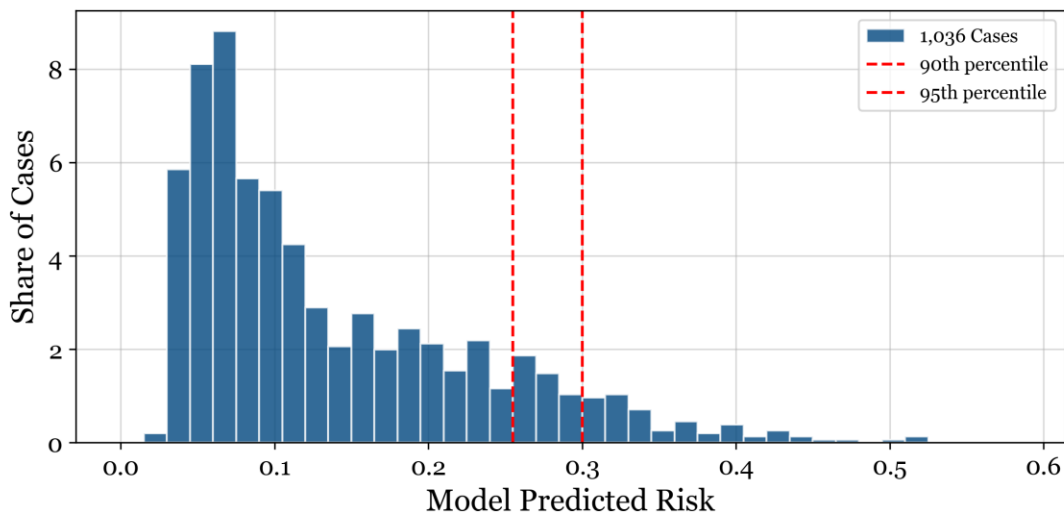
6. Email that list to Local QA team for review



Wrapper to be done
in "one call"

Sanity Checks along the Way

- Standardized logs and report are generated for each run:
 - Dates & times run, model versioning, merge rates
 - Features: extent of missingness and average values
 - Distribution of predicted risk
 - Comparison of high- and low-risk cases



	Highest Risk	Lowest Risk
Benefit Amount	\$522	\$288
Earned Income	\$2,148	\$166
Unearned Income	\$626	\$500
Self-employment Income	\$0	\$0
Shelter Expense	\$1,238	\$166

Implementation with Evaluation

How do we ensure that we're evaluating the model with respect to the metric which matters most?

Model Performance

Previous workshops, and most data science resources, focus on evaluating the statistical performance of models.

- AUC, precision-recall, FPR
- Cross-validated performance on historical data

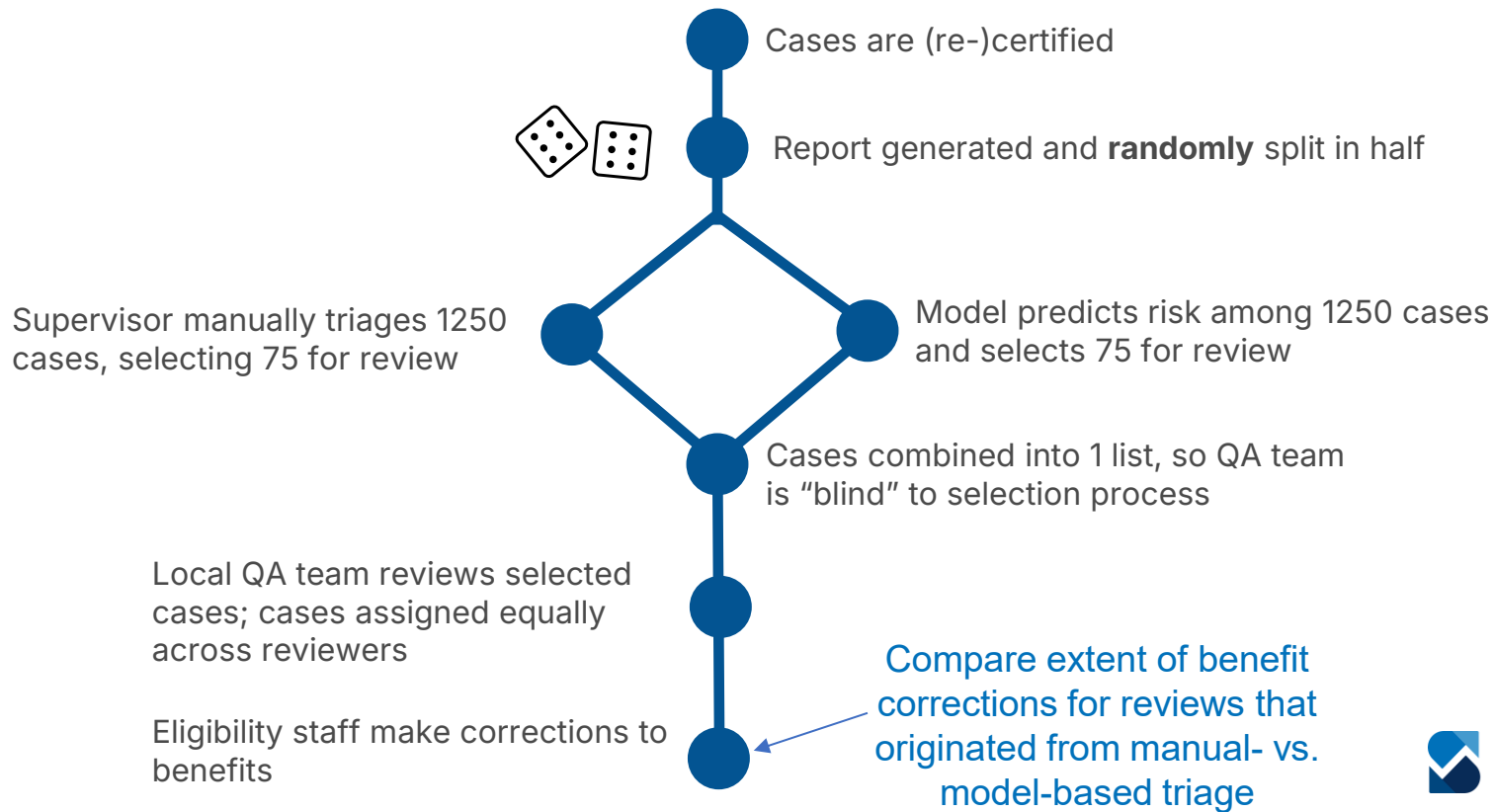
This is **necessary, but not sufficient** for the model to translate into a realized reduction in payment errors.

Model Use

QA $\neq$ QC. Processes are different and error outcomes we predict may not represent errors which can be identified fixed outside of detailed federal QC.

Data leakage, overfitting, and model drift. Small training sets create a reliance on older data, which may impact ability to predict errors **today**

Randomized Pilot of Model vs. Current Process



Thank You