# Claude Code for data science and visualization

Data Science and ML Engineering teams need sophisticated visualization tools to understand model performance, but building these tools often requires expertise in unfamiliar languages and frameworks. Claude Code enables these teams to build production-quality analytics dashboards without becoming full-stack developers.

## Main Claude Code use cases

**Building JavaScript/TypeScript dashboard apps**
Despite knowing "very little JavaScript and TypeScript," the team uses Claude Code to build entire React applications for visualizing RL model performance and training data. They give Claude control to write full applications from scratch, like a 5,000-line TypeScript app, without needing to understand the code themselves. This is critical because visualization apps are relatively low context and don't require understanding the entire monorepo, allowing rapid prototyping of tools to understand model performance during training and evaluations.

**Handling repetitive refactoring tasks**
When faced with merge conflicts or semi-complicated file refactoring that's too complex for editor macros but not large enough for major development effort, they use Claude Code like a "slot machine" - commit their state, let Claude work autonomously for 30 minutes, and either accept the solution or restart fresh if it doesn't work.

**Creating persistent analytics tools instead of throwaway notebooks**
Instead of building one-off Jupyter notebooks that get discarded, the team now has Claude build permanent React dashboards that can be reused across future model evaluations. This is important because understanding Claude's performance is "one of the most important things for the team" - they need to understand how models perform during training and evaluations, which "is actually non-trivial and simple tools can't get too much signal from looking at a single number go up."

**Zero-dependency task delegation**
For tasks in completely unfamiliar codebases or languages, they delegate entire implementation to Claude Code, leveraging its ability to gather context from the monorepo and execute tasks without their involvement in the actual coding process. This allows productivity in areas outside their expertise instead of spending time learning new technologies.

# Claude Code for data science and visualization

## Team impact

**Achieved 2-4x time savings**
Routine refactoring tasks that were tedious but manageable manually are now completed much faster.

**Built complex applications in unfamiliar languages**
Created 5,000-line TypeScript applications despite having minimal JavaScript/TypeScript experience.

**Shifted from throwaway to persistent tools**
Instead of disposable Jupyter notebooks, now building reusable React dashboards for model analysis.

**Direct model improvement insights**
Firsthand Claude Code experience informs development of better memory systems and UX improvements for future model iterations.

**Enabled visualization-driven decision making**
Better understanding of Claude's performance during training and evaluations through advanced data visualization tools.

## Top tips from the Data Science and ML Engineering teams

**Treat it like a slot machine**
Save your state before letting Claude work, let it run for 30 minutes, then either accept the result or start fresh rather than trying to wrestle with corrections. Starting over often has a higher success rate than trying to fix Claude's mistakes.

**Interrupt for simplicity when needed**
While supervising, don't hesitate to stop Claude and ask "why are you doing this? Try something simpler." The model tends toward more complex solutions by default but responds well to requests for simpler approaches.