Differences between GenAI development and traditional ML

Generative AI shares many core traits and foundational concepts with the waves of AI that have gone before, that focused on predictive and classification models; but there are some profound differences that cascade through to many differences in the activities and skills necessary to implement these new systems.

In a "traditional" ML/AI project, a Data Scientist works with a data set to train a model (or models) for the outcome they need - usually, to predict an attribute in the data, or to classify the data. This involves a number of activities:

- Identifying the model purpose and desired outcome (e.g. predicting equipment failure)
- Sourcing, loading and cleansing the data
- Performing "feature engineering " on the data (constructing features that can be used in a model, and will deliver effective results across a range of inputs)
- Training one or more models on the data, and tuning the hyperparameters for these models, to maximise performance
- Improving/updating feature engineering as necessary to improve model performance
- Incorporating human feedback/assessment into the model training (e.g. for classifiers)
- Determining the appropriate balance of model performance attributes (e.g. precision vs recall / false positives vs false negatives)

Once a model has been successfully trained, the Data Scientist or ML Engineer will undertake several more activities:

- Deploying the model to a production compute environment, fed by an engineered pipeline that productionises the data transformations created during model development
- Tuning implementation parameters/model selection to deliver the best mix of performance and cost to run
- Monitoring model performance and drift over time
- Maintaining a model library and version log
- Updating (retraining) the model periodically to maintain performance, and doing this in an auditable way (so that model predictions/classifications can be traced back to a particular version of the model)

However, a Generative AI project has a different workflow, as follows:

- Identifying system purpose and desired outcome
- Create a set of Q&A pairs that represent good performance against the use case
- Sourcing structured and unstructured data (as needed)
- Identifying/creating metadata for structured sources
- Indexing unstructured data
- Selecting a generative model (or models)
- Implementing a prompt workflow to capture input, integrate external data as necessary and generate output
- Create system prompts that will guide the performance of the generative models
- Testing Q&A pairs to assess performance
- Iterating system prompts and query handling to improve system performance
- Conducting UAT tests with users to assess real-world system performance and capture human feedback

As you can see, there is (kind of) a similar cycle across the two project types - they both start with a desired outcome (which it's important to be clear about, and have a way of measuring) and involve iterating towards a system that gets as close as possible to that outcome. But while, in a traditional ML context, this iteration is a relatively rapid, quantitative activity, founded on concrete steps such as changing parameter values or feature characteristics, in the GenAI world this is a much slower-wave iteration that involves multiple rewrites of system prompts and implementation of new architectural elements.

Additionally, while in the traditional ML world there is a (somewhat) clear distinction between a Data Scientist and data or ML engineers who will implement and monitor a model, this distinction is much less clear in the world of Generative AI - the optimisation process is a mixture of prompt tuning (a very non-deterministic activity) and architecture/data flow activities. Although it is possible to "train" a GenAI model (or at least fine-tune it) this is unlikely to be necessary or desirable for the majority of use cases - instead, prompt engineers work to guide and constrain an existing model's behaviour within acceptable guidelines.