AI & ML 101

For AEs

Scott Reed

July 2025

Agenda

Part 1: AI & ML Fundamentals

Part 2: Snowflake's View & Where to

Compete

2Part 1: AI & ML Fundamentals

Brief History of Machine Learning

Pre-2000: The Foundation Years Post-2000: The Data Explosion

● 1950s: Birth of Machine Learning

○ Term "machine learning" coined (IBM)

○ "Turing Test" to measure machine

intelligence

● 1960s-1970s: The AI Winter

○ Early chatbots and pattern recognition

(ELIZA)

○ Computers not strong enough

● 1980s-1990s: The Comeback

○ Neural network training breakthrough

○ IBM Deep Blue beats chess champion

(1997)

Smart ideas, but waiting for technology to catch

● 2000s: The Data Explosion

○ Proliferation of massive datasets

○ Computing power catches up

● 2010s: Deep Learning Dominance

○ AI beats humans at complex games

○ ML becomes part of everyday software

● 2020s: Generative AI Era

○ ChatGPT brings AI to mainstream

○ Anyone can create with AI tools

○ AI as common as smartphones

Technology caught up, data exploded, AI went

mainstream

up

4Let's clear this up…

Citation

 5Comparison to Similar Disciplines

Dimension Economics Statistics Machine Learning

Primary Goal

Explain & predict human/market

behavior to guide policy and

strategy

Draw reliable inferences about

populations from samples

Maximize predictive accuracy and

automate decisions

Theory-driven hypotheses (e.g.,

Hypothesis testing framework

Starting Point

supply & demand models)

with probability models

Data-driven pattern discovery

with minimal assumptions

Data & Tools

Published economic indicators;

Stata & MATLAB

Representative samples /

studies; R, SAS

Large, often messy,

high-dimensional datasets;

Python, ML Platforms, Notebooks

Key Output

Causal insights and policy

recommendations

Confidence intervals, p-values,

significance tests

Real-time predictions,

classification scores,

recommendations

Interpretability vs.

Accuracy

High emphasis on transparent

causal stories

High interpretability; statistical

significance required

Trades interpretability for higher

accuracy in complex models

 6Two Main Personas

Data Scientist / Research Scientist ● Frame business problems and conduct

exploratory data analysis (EDA)

● Build prototype models and validate

statistical insights

● Communicate findings through visualizations

presentations

● Handoff validated prototypes to engineering

teams

● Tools: SQL, python, advanced math, ML

platforms, notebooks, autoML

ML Engineer / ML Ops Engineer

● Refactor prototypes into scalable,

production-ready pipelines (CI/CD)

● Optimize model training, inference cost,

and latency performance

● Build APIs, containerize services, and

implement monitoring systems

● Handle real-time drift detection and

automated rollback systems

● Tools: Python, Git, Docker / Kubernetes,

Infrastructure as codeGlossary of Key Terms

Term Definition

Algorithm The mathematical "recipe" that tells a computer how to learn patterns from data

Model

The object that is created once an algorithm learns from data—what makes actual

predictions. This is a static object.

Model Training Teaching a model using historical data so it can make accurate predictions on new data

Model Scoring /

Inference

When the trained model makes predictions on new, real-world data—where business

value happens

Accuracy How often the model makes correct predictions, usually expressed as a percentage

Feature

Individual data points or columns the model uses to make decisions (age, purchase

history, etc.)

 Glossary of Key Terms (continued)

Term Definition

Feature

Engineering

Selecting and creating the most useful data inputs—often what separates good from great models

Label/Target The correct answer you want the model to predict ("will customer churn?", "optimal price?")

Hyperparameters Settings that control how the algorithm learns—like temperature and time when baking

Cross-Validation Testing technique to ensure the model will work on new data, not just training data

AutoML Technology that automates model-building, making ML accessible to non-experts

Drift Performance degradation over time—either data changes or accuracy drops, requiring retraining

A/B Testing Comparing two model versions to determine which performs better in real business conditions

 9Types of Machine Learning Techniques

Citation

 10Make it make

sense…

Supervised learning builds off

labels and the ultimate goal is

to build a model to predict

which “class” a new record

would fall into

Unsupervised learning takes

raw data without labels and

finds patterns in the data to

form logical “groupings”

Citation

 11Use Case to ML Technique Mapping

SHOPPER 360

OMNI PLANNING &

FORECASTING

MODERNIZING THE

RETAIL SUPPLY CHAIN

STORE OPERATIONAL

INTELLIGENCE

How to design a

customer centricity

How to forecast product

demand, optimize

How to design supply

chain visibility, improve

How to surface intraday

sales and operational

strategy?

pricing, and reduce

risk?

accuracy and prioritize

immediate actions?

performance metrics to

help support the store

teams understand

current sales patterns.

Customer Segmentation:

Unsupervised

Customer Lifetime Value

(CLV):

Supervised

Customer Churn:

Supervised

Demand Forecasting:

Supervised

Dynamic Pricing:

Supervised

Fraud Detection:

Supervised

RETAIL & CPG

DATA CLOUD

Inventory Optimization /

Demand Forecasting:

Supervised

Anomaly Detection in

Logistics:

Unsupervised

Shipment ETA:

Supervised

Market Basket

Analysis:

Unsupervised

Foot Traffic & Sales

Forecasting:

Supervised

RETAIL MEDIA

How to monetize data

through digital

advertising and shopper

insights?

Ad Targeting &

Click-Through Rate:

Supervised

Audience Segmentation:

Unsupervised

Personalized

Promotions:

SupervisedPart 2: Snowflake’s View &

Where to Compete

Model Lifecycle in Snowflake

UNIFIED GOVERNANCE

Develop

& Iterate

Orchestrate

Monitor

& Automate

Manage Deploy

& Serve

Develop

& Iterate

Orchestrate

& Automate

Manage Deploy

& Serve

Monitor

What is it? Why it Matters Discovery Questions

During this phase, Data Scientists

are doing EDA, feature

engineering, model building and

testing.

The bulk of ML use cases spend a

good amount of time in this stage.

This is a critical group to win over

and there is high compute

upside with daily, consistent

usage via notebooks, model

training, and feature engineering

pipelines.

Snowflake notebooks with

container runtime, distributed

APIs, along with git integration

makes us very competitive

How much time is spent moving,

sampling, or waiting for data versus

actually building and iterating on

models?

How do you ensure fresh,

production-quality data without

creating security risks or data

governance challenges?

When using open-source libraries

like Scikit-learn or PyTorch, how do

you manage the underlying

compute environments and

package dependencies?Develop

& Iterate

Orchestrate

& Automate

Manage Deploy

& Serve

Monitor

What is it? Why it Matters Discovery Questions

In this phase, ML engineers are

taking code developed by data

scientists and turnng it into

repeatable, production-grade

pipelines.

This involves running multi-step

data preparation and model

training workflows through

Snowpark and ML Jobs

(container runtime).

Simplifying the “hand-off” between

data scientists and ML engineers

creates stickiness for the whole

pipeline. We’re offering a

consistent environment with no

compute infrastructure to manage.

There is high compute upside in

getting these remote, executable

jobs (ML Jobs) that occur any time

data needs to be preprocessed

(new data comes in to be scored)

on container runtime.

How do you automate the model

preprocessing pipeline? What

separate tools are you using for

that orchestration?

How much of your engineering

team's time is spent maintaining

them versus building new ones?

How do you currently manage and

scale the compute resources for

your automated ML jobs?

Develop

& Iterate

Orchestrate

& Automate

Manage

Deploy

& Serve

Monitor

What is it? Why it Matters Discovery Questions

This is a centralized way to

discover, manage, and govern all
ML assets related to a model,
regardless of where it was
created.
Models are static in that the
training data, the hyperparameters
used, the model type, are all point
in time.
Model versions and their assets
for a particular experiment or
model run can vary greatly.
Connecting models to business
value is becoming more important
and less black box
Snowflake Model Registry and
ML Lineage are uniquely
positioned to add immediate
value.
Low compute upside - more
about mindshare of where models
are managed
How do you manage models that
are in production? Is it in a
centralized location or spread

across teams & git repositories?

How do you trace lineage back to

the exact features and data

version used to train it?

How do you track model versions

and who authorizes that?

Could a new teammate find,

understand, and manage model

versions?

Develop

& Iterate

Orchestrate

& Automate

Manage

Deploy

& Serve

Monitor

What is it? Why it Matters Discovery Questions

This is essentially the inference

step. It means moving a model

version into production to make

predictions on new data coming

in.

This model can be “served” to

accept new predictions in different ways - usually either in batch (make predictions overnight for the day’s data) or real time.

This is where all the magic happens in terms of translating AI into business value.

There is medium compute upside in owning where the inference occurs, what happens with the predictions themselves downstream, and the feature engineering pipeline.

Snowflake has two model serving options - warehouse for batch and SPCS for low latency. We also have a built-in feature store.

Do you currently have models in production today and what percentage are batch vs. near real time?

How do you ensure feature processing pipelines for new data you are scoring?

How do you manage the
underlying compute resources for
your deployed models (scale)?
How do you generate datasets for
training / inference?

Develop
& Iterate
Orchestrate
& Automate
Manage Deploy
& Serve
Monitor

What is it? Why it Matters Discovery Questions

Once a model is in production and
making predictions, you must
monitor to ensure it continues to
perform well.

Performing well generally means
that the chosen accuracy metric
remains consistent. Changes in
business processes or underlying
data (drift) can cause model
degradation.

The success of the use case

depends on the ability to successfully monitor models in production.

There is low compute upside in this phase, customer success in relation to business value is pertinent though.

Snowflake now has ML Observability & ML Explainability to round out a complete ML lifecycle.

How do you track model performance over time and what actions are triggered?

How do you detect and get alerted to model or data drift?

When a model's performance degrades, what is your process for troubleshooting the issue and triggering a retrain?

What is your process for auditing your ML models for compliance and explaining why a certain prediction was made?Compete: AWS

Strengths �� �� ��

● Sagemaker ML Platform is

“mature” in both capabilities

and product marketing

● No gaps in terms of their ML

features or tools

● Their ecosystem is sticky

and early Snowflake AI story

makes AWS seem like our

big brother.

Weaknesses

How can ❄ Win?

● It really isn’t a single platform -

while all the services exist, you

still have to stitch them

together yourself

● Emphasize architecture simplicity

● Reduce time to value by starting

and ending in Snowflake

● Architecture complexity to get

a model off the ground

● Forced data movement &

egress

● Land & Expand in the ML pipeline

○ If predictions come back to

Snowflake, why not use

our model registry for

inference (on externally

trained models)?Compete: Databricks

Strengths How can ❄ Win?

Weaknesses

�� �� ��

● “Industry Leading” in ML

mindshare

● Overwhelming and complex

depending on customer’s ML

maturity

● Grab their attention - find an

opening to showcase specific

capabilities

● Robust ML Ops framework

/ experimentation

● Spark experience required to

build pipelines, cluster

optimization

● Snowflake ML jobs for remote code

execution (chained notebooks

concept)

● Made for ML practitioners

(lots of horsepower)

● Lack of cost transparency to

track ML projects from start to

finish

● Position multi-modal offerings - AI

SQL capabilities combined with

Snowflake ML

Compete: Microsoft

Strengths Weaknesses

�� �� ��

● Really good for customers

who are all in on the

Microsoft ecosystem

● They are effectively single cloud

when it comes to Azure ML

● Tightly integrated to all

Microsoft services and/or

databricks

● For customers leveraging both

Databricks ML and Azure ML, the

line in the sand on separation of

duties hurts their overall

messaging

● Mature ML platform and

great for practitioners

● If using delta lake this can add

complexity for teams and

governance challenges

How can ❄ Win?

● If delta lake users and proud of

“open source”, challenge them

to look at their Azure centric

stack

● Pitch multi-cloud

● Find pain points / governance

challenges in bronze layer

● Land & Expand playKey Takeaways & Call to Action

THANK YOU