Applied Machine Learning for Business Analytics

Lecture 12: Why do ML Projects Fail in Business

Lecturer: Zhao Rui

Logistics

- Thanks for your participation in the Kaggle Competition
- We have two quiz for attendance check (4 points)
  - Even if you miss it, you will get 6 points for attendance
  - Scores would be released before next Friday
- Group project would be due @ Friday, May 1st
  - Naming of files!
  - Word and latex templates have been provided

Agenda

1. What is Skills
2. Courser Summary

LLMs Are Stateless Text Machines

An LLM is a function — nothing more:

```
f(context_window) → next_tokens
```

- No persistent memory across sessions
- No ability to act on the world
- No knowledge of your specific workflows or domain
- Every capability we associate with modern AI agents is an engineering layer built on top
  - Memory, tool use, structured outputs, domain expertise
  - Agent skills are one such layer

To understand why Skills exist and how they work, we need to see the full stack.

The Capability Stack: From Prompts to Skills

Layer	What It Is	What Problem It Solves
Prompt	Raw text input to the model	Tells the model what you want right now
System Prompt	Persistent instructions prepended to every turn	Gives the model a persona and rules across a conversation
Tool Calling	Structured interface for the model to invoke functions	Lets the model act on the world (search, compute, API calls)
MCP	Standardized protocol for connecting to external data/services	Lets tools be portable and composable across models and platforms
Agent Skills ★	Packaged instruction sets with progressive loading	Gives the model domain expertise on demand without blowing up context

Skills don't replace prompts, tool calling, or MCP — they orchestrate and leverage them.

What is an Agent Skill, Really

Strip away the packaging: a Skill is a structured prompt that loads itself on demand.

- A documentation file (SKILL.md) that an LLM consults at inference time
- Encodes rules, formats, examples, and logic for a specific task domain
- Uses progressive disclosure to avoid wasting context tokens on irrelevant instructions
- If you've ever written a long system prompt — "when user asks about X, follow these 47 rules..." — you've built a primitive skill

Prompts tell the model what to do once.
Skills teach the model how to do something repeatedly and correctly.
A prompt is ephemeral. A skill is reusable institutional knowledge.

Origin: Claude-specific feature, open-sourced by Anthropic on December 18, 2025 for cross-platform compatibility.

Anatomy of a Skill: Structure & Metadata

Folder structure

```
my-skill/
├── SKILL.md        # The skill itself
└── resources/      # Optional: docs, scripts
```

Metadata header (YAML)

```
---
name: weekly-report
description: >
  Generates a formatted weekly treasury
  report from BigQuery data. Use when
  the user asks for weekly ops summaries,
  treasury briefs, or end-of-week reporting.
---
```

Why the description is critical

- The model never reads all skills upfront
- It reads only descriptions (~50 tokens each) to decide which skill to activate
- ⚠️ **Bad description = skill never fires** ⚠️
- Analogous to a tool's description field in function calling

The description is the skill's "trigger function."
Be specific about when it should activate. Include keywords the user is likely to say.

Anatomy of a Skill: The Instruction Body

Everything below the YAML header — the full playbook for a task domain.

What goes in the body

- Rules and constraints
  - "Always use ISO 8601 dates"
  - "Never include PII"
- Output format specifications
  - JSON schema, markdown templates, table structure
- Few-shot examples
  - Sample input/output pairs that anchor behavior
- Decision logic
  - "If user provides a ticker, fetch live price; otherwise use cached daily close"
- Resource pointers
  - "Read resources/handbook.pdf for the glossary"

Progressive Disclosure: The Problem It Solves

The naive approach

- Agent has access to 50 skills
- Load every skill's full instructions into context at all times
- Result:
  - Tens of thousands of tokens burned before the user says anything
  - Wasteful and expensive
  - Long conversations exceed context limits
  - Irrelevant instructions can confuse the model

Analogies from computing

- Lazy loading in software engineering
  - Don't load a module until it's actually needed
- Attention sparsity in transformer architecture
  - Don't compute what you don't need
- Context engineering
  - The craft of managing what goes into the context window
  - Skills are a concrete implementation of this idea

Progressive disclosure is a context management strategy: don't load what you don't need.

Progressive Disclosure: Three-Layer Loading

Layer 1: METADATA [Always loaded]

Names + descriptions only (~50 tokens each). The model scans this like a table of contents to find the right skill.

▼ model decides skill is relevant ▼

Layer 2: INSTRUCTIONS [On-demand]

Full SKILL.md body loaded only when the model decides this skill applies. Rules, examples, decision logic.

▼ layer 2 calls for it ▼

Layer 3: RESOURCES [Deep on-demand]

Reference docs, scripts, templates. Loaded only if Layer 2 explicitly requires them.

Token cost at each layer

Layer	Tokens (typical)	When paid
L1 Metadata × 50 skills	~2,500	Every request
L2 Instructions (1 skill)	500–3,000	When skill fires
L3 Resources	0–10,000+	When explicitly needed

Compare: loading all 50 skills' full instructions naively = **25,000–150,000 tokens** on every single request.

Resources: Reading vs Executing

Dimension	References (Read into Context)	Scripts (Execute Outside Context)
What it is	Documents the model reads e.g., financial glossary, style guide, SOP	Code the model runs e.g., Python script querying BigQuery
How it works	Content enters the context window directly	Model sees only the interface + output; script body stays outside
Token cost	⚠️ **Consumes tokens proportional to document size**	✓ **Near-zero cost — script can be thousands of lines**
Analogy	Retrieval in RAG — inject knowledge at inference time	Tool calling — delegate computation to external process
When to use	Small content; model needs to reason over it directly	Large/structured data; database, API, complex computation

Design rule: if a resource is large and structured, make it a script — not a reference.

Design Principles for Good Skills

Write the description like a trigger condition. Be specific about when it should activate. Include keywords the user is likely to say. Bad description = skill never fires.

Show, don't tell. Few-shot examples in the instruction body are the single most effective way to control output format. One concrete example beats a paragraph of abstract rules.

Separate knowledge from computation. If something can be a script (data fetching, formatting, file generation), make it a script. Reserve the instruction body for reasoning logic the model needs in-context.

Keep token budgets in mind. A skill that loads a 10,000-token reference document on every activation is expensive. Ask: does the model need to reason over all of this, or can it be summarized or chunked?

Version and iterate. Skills are text files. Put them in git. Track what works. A skill is a living artifact, not a one-time prompt.

2. Course Summary

Schedule

Class Venue: BIZ2-0511

Date	Topic	Content	Assignment
Fri 01/16	Introduction	Link	N.A.
Fri 01/23	Text Representations: BoW to Word2Vec	Link	N.A.
Fri 01/30	Transformers	Link	Cancelled. Make-up: 02/06 & 02/13
Fri 02/06	LLM Fundamentals	Link	Assignment I Out
Fri 02/13	Training & Scaling LLMs	Link	Assignment I Due
Fri 02/20	Inference & Reasoning	Link	Assignment II Out
Fri 02/27	Recess Week	N.A.	Proposal & Assignment II Due
Fri 03/06	LLM Applications	Link	Assignment III Out
Fri 03/13	Agent: Tools	Link	Assignment III Due
Fri 03/20	Agent: Thinking Patterns	Link	Mini Project Starts
Fri 03/27	Agent: Memory	Link	N.A.
Fri 04/03	Good Friday	N.A.	N.A.
Fri 04/10	LLM & Agent Evaluation	Link	Mini Project Competition
Fri 04/17	Why ML&LLM Projects Fail	Link	N.A.

Thank You

- Immense thanks to Dingyu and Ding Yang !!!
- Enjoy having all of **you** in BT5153 this year. Appreciate your hard work!