

Development of Housing Price Trends Visualization App using Agents and Korean Open-Dataset

Jack Yang

Tandon School of Engineering
New York University
New York, United States
jy4367@nyu.edu

Rolando Sanchez III

College of Letter & Sciences
Univeristy of California, Santa
Barbara
Santa Barbara, United States
rolandosanchez@ucsb.edu

Luke Lee

Lake Forest Academy
Lake Forest, United States
luke.lee@students.lfanet.org

Rakil Kim

UCLA Henry Samueli School of
Engineering and Applied Science
University of California, Los
Angeles
Los Angeles, United States
rakilkim0216@gmail.com

Alan Dowan Kim

Cupertino High School
Cupertino, United States
alan.dowan.kim@gmail.com

SeungEun Eo

College of Human Ecology
Yonsei University
Seoul, South Korea
eosveunvg@gmail.com

Youngchae Kim

College of Human Ecology
Yonsei University
Seoul, South Korea
yc20807kim@gmail.com

Jin-Kook Lee

College of Human Ecology
Yonsei University
Seoul, South Korea
leejinkook@yonsei.ac.kr

Abstract—This paper presents a platform that enables users to query, compare, generate, and visualize housing price trends using Korean Open-Dataset and AI agents. The system leverages LangGraph, the ReAct framework, and Model Context Protocol (MCP), to create an intuitive platform for non-expert users. A key finding is the system’s ability to create temporary agents from user data in real time. This seamless integration enables comparison of heterogenous datasets (User uploaded datasets in various units and Korean dataset in Price indices) by scaling values for visualization. Through multiple different scenarios, we demonstrate how this system outperforms standard chatbots and traditional methods in accuracy and usability. The results demonstrate that our agentic approach offers a more intuitive and powerful tool than traditional methods or general-purpose LLMs, providing significant support for informed decision-making.

Keywords—*Agentic AI, Large Language Models (LLM), Housing Market Analysis, Natural Language Interface (NLI), Multi-Agent Systems, Data Visualization, Human-AI Interaction*

I. INTRODUCTION

The housing market is a complex domain with drastic impacts on the economy and people’s everyday lives. People plan their lives around changes in housing values, with housing equity making up over 60% of an average Korean household’s total net worth [1]. A home is typically the largest and riskiest investment an individual makes, greatly influencing their financial stability, retirement planning, and quality of life [2]. Consequently, timely and simple access to clear and accurate housing market data is critical for informed decision making [3].

However, current methods of accessing and interpreting housing data present significant barriers and limitations for most users, such as cost and technical expertise [4]. While chatbot-based natural language interface (NLI) solutions partially lower these barriers, they often fail to handle complex, multi-step queries and lack dynamic adaptability [5]. As a result, users are still restricted in how deeply and easily they can engage with housing data.

This research addresses these challenges by developing an intuitive, user-friendly system powered by agentic AI—a paradigm where systems autonomously pursue complex goals, dynamically adapt to environmental inputs, and interact with multimodal data in real time [6], [7]. Specifically, we develop a platform designed to simplify the querying, visualizing, and comparing of housing data for the Korean market. Our proposed solution utilizes a multi-agent workflow that simplifies complex data analysis, making sophisticated market insights accessible to a broader range of stakeholders, including prospective homebuyers, policy analysts, and researchers

II. BACKGROUND

In recent years, Chatbot-Based Natural Language Interfaces (NLI) has emerged as a potential solution for improving data visualizing and querying. However, as shown in Kavas et al. [5], these NLI are far from sufficient, with the authors identifying key limitations such as the lack of support for high level follow up queries, sophisticated interaction methods, and ambiguities. These systems are limited in their flexibility, resulting in a shallow and rigid user experience, requiring tailored prompts and extra user input. This makes current NLI unsuited for the kind of iterative, comparative, and analytical reasoning needed in the complex domain of housing economics.

A. Evolution from Reactive Systems to Agentic Intelligence

Recent advancements in artificial intelligence have culminated in the emergence of Agentic AI—a paradigm where systems autonomously pursue complex goals, dynamically adapt to environmental inputs, and interact with multimodal data in real time [6], [7]. Enabled by large language models (LLMs), reinforcement learning, and memory-based planning, these systems form an evolution beyond conventional task-specific AI toward goal-driven, self-directed agents [8], [6].

This technical evolution reflects a broader shift in how intentionality and control are conceptualized in intelligent systems. The term “agentic” was originally used in psychology by Albert Bandura, who described humans as proactive and self-regulating beings, capable of exercising control over their development and actions [9]. As artificial intelligence progressed, especially with the integration of reinforcement learning and foundation models, this notion of agency was extended to machines capable of autonomous goal formation and adaptive decision-making [6], [7]. The result was a new class of AI agents defined not merely by reactivity but by intentional behavior and self-directed learning.

In particular, Sapkota, Roumeliotis, and Karkee argue that Agentic AI marks a departure from traditional agents by exhibiting high-level cognitive coordination, goal decomposition, and the capacity to dynamically manage sub-tasks in pursuit of shared objectives. They define these systems as possessing “higher-order cognition and self-organized goal pursuit capabilities”—a functional expansion beyond existing AI frameworks [7].

B. Core Technologies Enabling Agentic AI

Agentic AI integrates several foundational components. Reinforcement learning enables long-term planning via cumulative reward optimization. Adaptive control and meta-learning mechanisms help agents recalibrate strategies as data conditions change, while persistent memory modules retain cross-session context for coherent goal pursuit [5], [6].

Multimodal interfaces further equip these agents to interpret structured and unstructured data across formats—spreadsheets, images, APIs—using task decomposition and agent orchestration techniques [7], [10]. One agent might analyze regional statistics, another might generate visual outputs, and together they manage dynamic workflows with minimal oversight [11], [12].

C. Technologies Enabling our Development

This paper proposes an agentic workflow with multi agent collaboration for analysis, comparison and visualization of Korean housing price datasets. An agent in this context is an autonomous LLM module that processes inputs and acts to achieve specific goals. Agents are designed to be goal oriented and capable of planning and making decisions. Each agent is equipped with the Reasoning and Acting (ReAct) framework, which allows the agent to refine its reasoning and actions in an iterative loop, using external tools and trying a new approach when it deems necessary [13]. These agents are coordinated through LangGraph, a Python framework for building multi-agent workflows as graphs. In LangGraph, each functional unit is represented as a node. Nodes are connected through edges. Types of nodes include LLM-based agent nodes, and tools nodes, which contain external tools the LLM-based agent nodes can call. To generate visualization, the solution uses a Model Context Protocol (MCP) server, specifically AntV’s MCP Server Chart. MCP is an open standard protocol that standardizes how AI applications interact with external tools. This agentic architecture, combined with the ReAct framework

and MCP, allows for flexibility and adaptability while reducing the likelihood of AI hallucinations.

III. IMPLEMENTING AGENTIC ARCHITECTURE

The solution implements an agentic AI architecture with the ReAct framework powered by LangGraph. The LangGraph workflow functions as a dynamic loop directed by the main agent. The main agent is responsible for interpreting user inputs, determining appropriate tools to call, creating sub-inputs for other agents based on initial user inputs, and managing the overall decision flow. Outputs from external tools or other agents are sent back to the main agent, who decides whether the output fulfills the user’s request. If the output is incorrect or insufficient, the main agent adjusts its plan, calling other tools or modifying the inputs for its tool calls. This repeats until the main agent is satisfied with the output. Figure 1 showcases a simple diagram explaining the system.

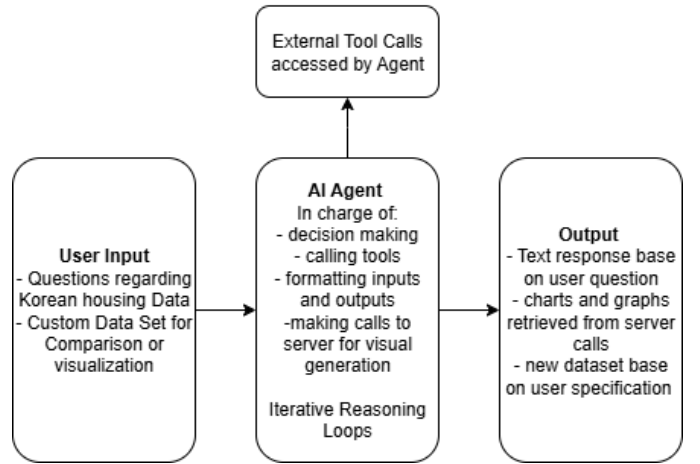


Fig. 1: High-level System Overview Diagram

The architecture consists of multiple specialized agents, each built using the `create_csv_agent` function in the LangChain experimental toolkit. Two core agents were developed using said functions, each handling a different aspect of the Korean housing market. Both agents are using ChatGPT’s 4.1-mini LLM model, selected for its low cost per token and relatively high performance. OpenAI’s ChatGPT models were the only LLMs tested that were able to successfully query information from the CSV files. The first agent was given access to data regarding Korea’s jeonse housing market, a type of long term lease unique to the country where renters pay a deposit around 50% to 80% of the property’s value. The second agent was given access to data regarding Korea’s home sales. Both datasets were obtained from the Korean Statistical Information Service (KOSIS) and the change in price is recorded through a price index, the standard form of measurement from the Korea Real Estate Board, with June 2021 scaled to 100. Korean housing data are subdivided into housing type, (Apartment, Row Houses, and Detached Houses) and scale (ranging from 1 to 6 with 1 begin the smallest in size). Both agents were initialized with a domain specific system message that included an overview of the agent’s role and basic metadata about the dataset. The metadata was included in the system message as an additional guardrail and can be omitted, as the agents are able to query the datasets

for metadata independently. During the setup of the LangGraph workflow, both agents would be converted to a tool node, allowing for the main agent to call them.

Two more auxiliary tools were integrated into the LangGraph workflow. The first was a create csv file tool which the main agent can call after querying information from either the user uploaded data set or from the jeonse and housing agent. The tool accepts a Json payload and outputs a dataset based on the given payload. The tool contains a docstring that informs the main agent of the tool's purpose and the desired format of the payload.

The second tool is a generate chart tool, which makes a call to AntV's MCP Server Chart and outputs a link to the generated chart. The transport, the mechanism that allows for communication between the agent and the server, is Server-Sent Events (SSE), which allows the streaming of multiple server messages. SSE was chosen for its efficiency in streaming real-time responses to the user interface, ensuring low-latency updates

The workflow is constructed using LangGraph. Each agent and tool is represented by a node and connected using edges. The jeonse agent and housing agent were both converted to tool nodes using LangGraph's `as_tool` function. The two agents, along with the auxiliary tools, are integrated into the workflow using a tools node. The execution graph begins at a start node, which is connected to the main agent node using an edge. This central agent interprets the user's input and determines the appropriate course of action. The flow of action then goes through a conditional edge, implemented using LangGraph's `add_conditional_edges` function. The conditional edge has two routes. If the main agent calls a tool, the edge routes to the tool node, which upon generating an output, will route back to the main agent. The main agent can then decide to call another tool, which will trigger the edge to route to the tool node again. When the main agent completes all its tool calls, the flow of action will then be routed to the end node. Figure 2 showcases the custom agentic workflow developed in this paper.

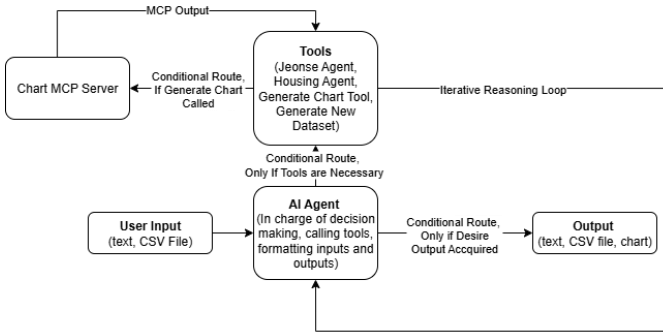


Fig. 2: Custom Agentic Workflow

The system can also accept external datasets and adjust according. When a user upload a dataset, the workflow is dynamically augmented using LangChain and LangGraph's `create_csv_agent` and `as_tool` function. The system creates a new agent wrapped as a tool, similar to the process of initializing the Jeonse and housing agent, and appends it to the current tool node. This allows the main agent to use the tool alongside other tools, such as the Jeonse and housing agent, enabling for the

comparison and visualization of the user uploaded data with Korean housing price data.

Finally, the user interface (UI) was designed as a chat interface, offering an intuitive experience for interacting with the agentic AI system. The layout includes familiar chat elements such as message bubbles, typing indicators, timestamped responses, and a clear chat button. The interface also includes support for user uploaded dataset. Figure 3 showcases the user interface.

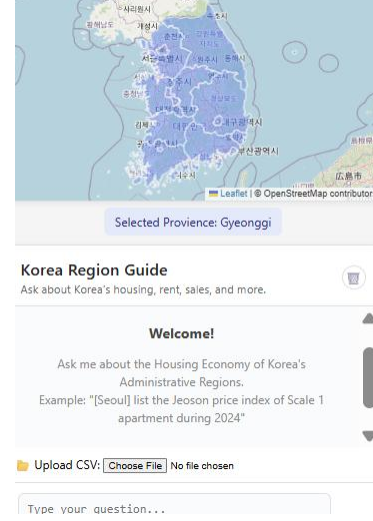


Fig. 3: Chatbot UI with Interactive Map and File Upload Tool

IV. DEMONSTRATION

To demonstrate the capabilities of the proposed agentic AI solution, we present three user scenarios that showcase its ability to handle natural language queries, generate visuals, and integrate user provided datasets. The domain is the Korean housing market.

A. Scenario 1: Domain-Specific Querying

A user begins by selecting a region in the interactive South Korean map. Then the user can ask a question regarding the selected region's housing prices such as, "What is the average housing price index of apartments during 2024?" If no region is selected, the agent handles the ambiguity by assuming that the user wants data regarding the entirety of the nation. Upon submitting this message via the web UI, the backend constructs a LangGraph graph, instantizing all tools and agents. The user message gets routed to the main agent who makes the decision of which tool to call, in this case being the housing price agent. The housing agent will go through its own iterative reasoning loop based on the inputs provided by the main agent. The housing agent uses Python REPL, a LangChain tool with the ability to execute code, to query the dataset. If the first query attempt is not successful, the housing agent would reflect on the issue and adjust its code accordingly. Figure 4 provides a snippet of the housing agent's thought process. The first action returns a string saying "No data available for Apartments in Seoul at Scale 1" which the agent recognizes is not the desired solution, leading it to enter a new thought chain stating "The 'python_repl_ast' tool is the correct tool to use for Python

commands. I will retry the action with the correct tool.” The final output from the housing price agent is routed back to the main agent, who will decide whether the output is sufficient. If sufficient, the main agent would format this result into a natural language response and display it in the chat window, as seen in Figure 5. The user can ask ambiguous follow up questions, which the agent will interpret using past conversations (Figure 5).

```
Thought: The dataframe sample provided shows data for "TheWholeCountry" region, but the question asks for apartments in Seoul for scale 1 during 2024. I need to filter the dataframe for Type(1) as "Apartments", Region(1) as "Seoul", and Scale(1) as "Scale1". Then I will select columns corresponding to the months in 2024 (from "2024.01" to "2024.12").

Action: [python_repl_ast]
Action Input: df_filtered = df[(df['Type(1)'] == 'Apartments') & (df['Region(1)'] == 'Seoul') & (df['Scale(1)'] == 'Scale1')]
months_2024 = [f'2024.{str(i).zfill(2)}' for i in range(1, 13)]
df_filtered[months_2024] if not df_filtered.empty else "No data available for Apartments in Seoul at Scale 1"
[python_repl_ast] is not a valid tool, try one of [python_repl_ast].Thought: The "python_repl_ast" tool is the correct tool to use for Python commands. I will retry the action with the correct tool.

Action: python_repl_ast
Action Input: df_filtered = df[(df['Type(1)'] == 'Apartments') & (df['Region(1)'] == 'Seoul') & (df['Scale(1)'] == 'Scale1')]
months_2024 = [f'2024.{str(i).zfill(2)}' for i in range(1, 13)]
result = df_filtered[months_2024] if not df_filtered.empty else "No data available for Apartments in Seoul at Scale 1"
result
2024.01 2024.02 2024.03 2024.04 2024.05 2024.06 2024.07 2024.08 2024.09 2024.10 2024.11 2024.12
6 92.7 92.6 92.4 92.5 92.6 92.8 93.2 93.7 93.9 94.0 94.1
Thought: I have found the housing price index data for apartments in Seoul at Scale 1 for each month of 2024 in the dataframe. The data is in the row indexed 6 for the months 2024.01 to 2024.12.
```

Fig. 4: Housing Agentic Iterative Reasoning

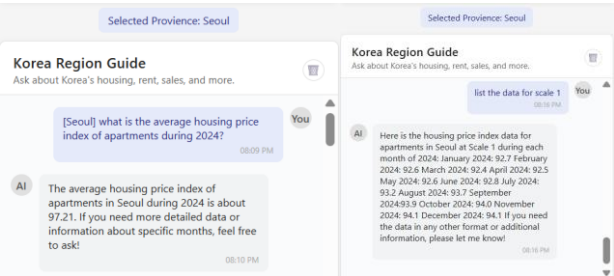


Fig. 5: Domain Specific Querying

By contrast, a standard Google search or querying a generic LLM chatbot would require much more specificity and would be unable to return an accurate result. Figure 6 showcases the differing result quality from Google Search’s built in Gemini model and OpenAI’s GPT-4o model. Both were unable to provide an answer or were inaccurate in interpreting the user’s message.

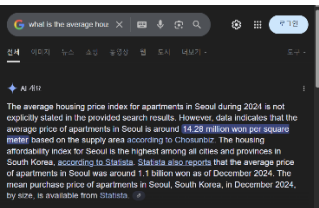


Fig. 6: Standard LLM and Search Output Comparison

B. Scenario 2: Comparative Analysis and Chart Generation

The user can also compare and visualize price data from different data sets. For instance, if the user request “Compare jeonse and home sale price indices for a scale 1 apartment in Seoul during 2024 and show me a chart”, the main agent reasons that the jeonse agent and home sales agent must be invoked, followed by the call to the Chart MCP Server. Using the outputs from the jeonse agent and home sales agent, the

main agent assembles a Json payload specifying a dual line chart and invokes the Chart MCP server using said payload. The output would then be returned to the user as a chart link embedded in the chat. Figure 7 displays the text output and its corresponding chart.

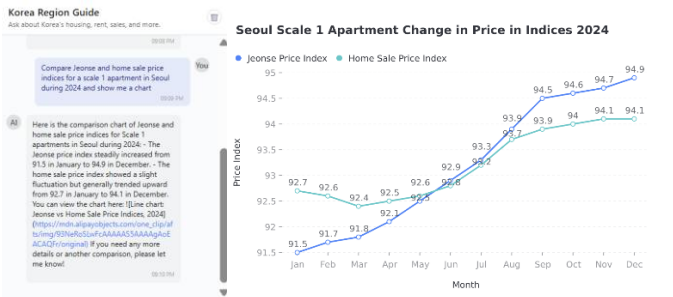


Fig. 7: Comparison and Visualization Demonstration

Performing the same comparative analysis via Google would require separate searches for each index, manual data exporting, and external chart building skills. Similarly, OpenAI’s GPT-4o model was unable to gather proper data or generate a chart, as the model relies on scraping the internet for data (Figure 8).

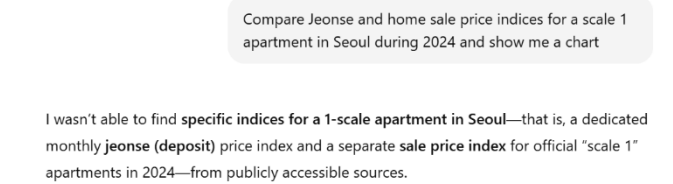


Fig. 8: Standard LLM Comparison

C. Scenario 3: User-Uploaded Data Integration

To demonstrate extensibility, we will use user uploaded data containing raw housing prices in New York City, obtained via StreetEasy’s open data dashboard. This differs from the Seoul data which is a pricing index. The agent is able to perform complex multi-step queries and scale the data for a graph visualization. Figure 9 showcases the ability to integrate user uploaded data with preloaded tools, such as the jeonse agent and generate chart tool. The user asks for the median asking price in Bensonhurst, a neighborhood in NYC, and the jeonse price index of a apartment in Seoul during 2024. The agent recognizes that the first request can be found in the user uploaded file and the second request can be found in the jeonse agent. The user then followed up with another request to generate a graph comparing the two lists of values. Note that the lists are in different units, raw dollar amount and an index, but the agent is able to scale the values in the graph for meaningful comparison.

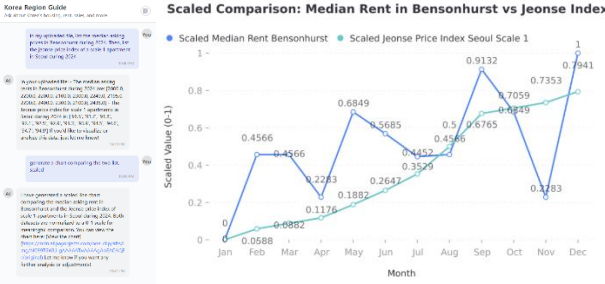


Fig. 9: User-Uploaded Data Demonstration

D. Scenario 4: Custom Dataset Creation

In this scenario, users can generate a new dataset tailored to their needs and based on data from the existing tools and user uploaded files. This feature is especially useful when the user has precise criteria in mind, such as region or time, and wishes to export a refined subset of data. This empowers users to transform the system from a data retrieval tool to a data generation engine.

For instance, Figure 10 showcases the generation of a file with pricing data of just Seoul apartments. This entire process was completed in just one text prompt. Replicating this workflow with traditional tools would either require proficiency in query languages or manual filtering. A generic LLM would at best provide steps to complete the task and would be unable to return a proper output without significant help and extra data.

Type	Region	Scale	2021.06	2021.07	2021.08	2024.11	2024.12	2025.01	2025.02	2025.03
Apartment Seoul	Scale1	100	100.2	100.7	94.6	94.7	94.9	95	95.1	95.5
Apartment Seoul	Scale2	100	100.8	101.7	92.3	92.5	92.5	92.5	92.6	92.8
Apartment Seoul	Scale3	100	100.6	101.3	90.1	90.3	90.3	90.3	90.3	90.5
Apartment Seoul	Scale4	100	100.7	101.5	89.1	89.3	89.5	89.6	89.6	89.8
Apartment Seoul	Scale5	100	100.7	101.3	89.2	89.5	89.5	89.4	89.5	89.7
Apartment Seoul	Scale6	100	100.9	101.4	92.5	92.9	93	93	93.1	93.2
Apartment Seoul	Scale1	100	100.8	101.8	94	94.1	94.1	94	93.9	94.1
Apartment Seoul	Scale2	100	101	102.1	95.5	95.6	95.7	95.6	95.7	96.3
Apartment Seoul	Scale3	100	100.7	101.6	88.7	89	89.1	89.1	89.4	102.2
Apartment Seoul	Scale4	100	101	101.7	101.9	102.2	102.4	102.4	103	104.5
Apartment Seoul	Scale5	100	100.8	101.5	100.8	101	101.2	101.2	101.6	102.5
Apartment Seoul	Scale6	100	100.7	101.3	106.1	106.5	106.8	107	107.5	109

Fig. 10: Custom Dataset Demonstration

V. CONCLUSION

This research proposed an innovative solution to accessing housing data through an agentic AI platform, designed to empower users with little to no technical expertise. Compared to traditional methods, which often require SQL. Compared to traditional methods that rely on SQL-based queries, our system leverages a multi-agent workflow for visualization and comparison. In contrast to general purpose LLM, our system is optimized for the Korean housing domain and supports real time interactions between specialized agents, external tools, and MCP servers.

Future work can explore applying this architecture to other domains and expanding the data available to the agents. Additionally, expanding the tool list to include maps may be useful in visualizing the difference between regions. Although the agents are optimized for their respective dataset, their

performance remains constrained by the quality and recency of the data. Improvement can be made by incorporating APIs that update the dataset on a consistent basis. Additionally, the ReAct framework can be very costly, especially in cases where the agent continuously tries and fails to come up with a solution. Misinterpretation of prompts can occasionally result in undesired outputs. For example, the agent would occasionally return the average price index when asked to list each individual value. By continuing to address these challenges, we envision the evolution of this platform into a truly collaborative and intelligent decision-support partner, fundamentally changing how stakeholders interact with complex urban data.

ACKNOWLEDGEMENT

This research was carried out within the Space Management and Design IT Lab, under the supervision of Professor Jin-Kook Lee, as part of the YISS undergraduate research internship program at Yonsei University.

This work is supported in 2025 by the Korea Agency for Infrastructure Technology Advancement (KAIA) grant funded by the Ministry of Land, Infrastructure and Transport (Grant RS2021-KA163269).

REFERENCES

- [1] S.-W. (Stanley) Cho, "Household wealth accumulation and portfolio choices in Korea," *Journal of Housing Economics*, vol. 19, no. 1, pp. 13–25, Mar. 2010.
- [2] "Housing wealth and retirement savings: enhancing financial security for older Americans," *Harvard.edu*, Sep. 07, 2005.
- [3] J. Ge and T. Harfield, "The quality of data and data availability for property research," *opus.lib.uts.edu.au*, Jan. 2007.
- [4] M. Bara, "Accessibility barriers in multi-terabyte public datasets: the gap between promise and practice," *arXiv.org*, 2025.
- [5] Ecem Kavaz, A. Puig, and I. Rodriguez, "Chatbot-based natural language interfaces for data visualisation: a scoping review," vol. 13, no. 12, pp. 7025–7025, Jun. 2023.
- [6] D. B. Acharya, K. Kuppan, and B. Divya, "Agentic AI: autonomous intelligence for complex goals—a comprehensive survey," *IEEE Access*, vol. 13, pp. 18911–18934, 2025.
- [7] R. Sapkota, K. I. Roumeliotis, and M. Karkee, "AI agents vs. agentic AI: a conceptual taxonomy, applications and challenges," *arXiv preprint arXiv:2505.10468*, 2025.
- [8] A. Zeng et al., "AgentTuning: enabling generalized agent abilities for LLMs," in *Findings of the Assoc. for Comput. Linguistics (ACL)*, 2024, pp. 3053–3077.
- [9] A. Bandura, "Social cognitive theory: an agentic perspective," *Annu. Rev. Psychol.*, vol. 52, no. 1, pp. 1–26, 2001.
- [10] Y. Shavit et al., "Practices for governing agentic AI systems," *OpenAI Research Paper*, Dec. 2023.
- [11] Y. Matsuo et al., "Deep learning, reinforcement learning, and world models," *Neural Networks*, vol. 152, pp. 267–275, 2022.
- [12] S. Kapoor et al., "AI agents that matter," *arXiv preprint arXiv:2407.01502*, 2024.
- [13] S. Yao et al., "ReAct: synergizing reasoning and acting in language models," *arXiv.org*, Mar. 09, 2023.