

TrustAsset

*Smart Platform for Buying, Selling, and Renting
Cars and Real Estate (Mobile & Website)*





Presented by:

Muhammad Zaid Al-Nahas

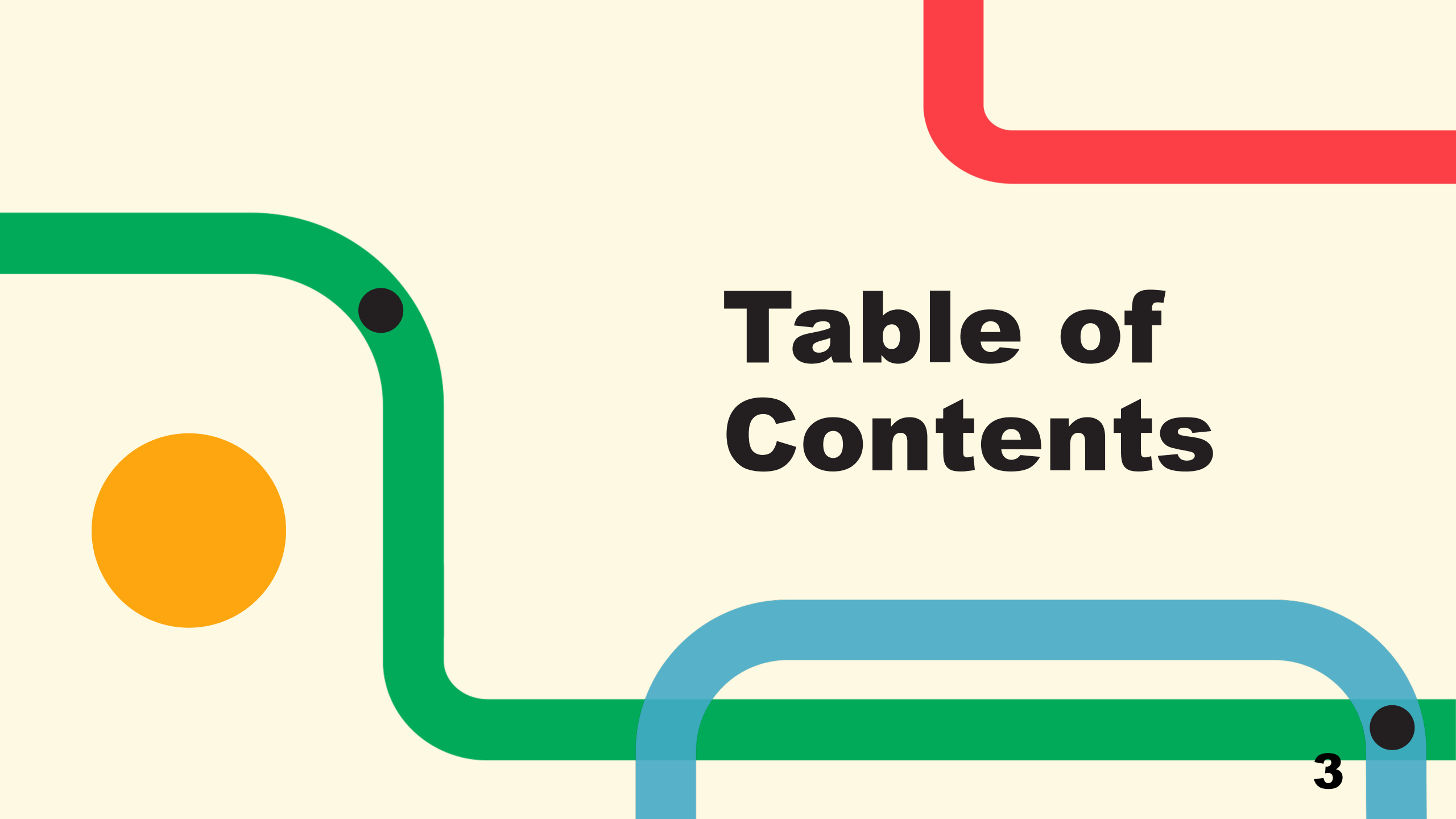
A decorative graphic on the left side of the page. It features a green line that starts from the left, curves down, and then continues horizontally. A blue line starts from the bottom, curves up, and then continues horizontally, overlapping the green line. A red line starts from the top right and curves down. An orange circle is positioned to the left of the green line. There are two black dots: one on the green line and one on the blue line.

Table of Contents

Data Preparation for Modeling



Dataset Overview

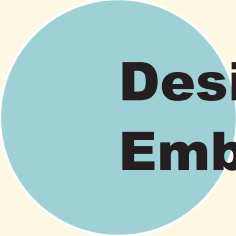
Data Exploration

Data Preprocessing

Final Review

Exporting Processed Data

Content-Based Recommendation System (Embedding Approach)



Design and Train Neural Network for Embeddings



Generate Item Embeddings



Calculate Embedding Similarity



Create Embedding Similarity Lookup



Generate Synthetic User Interaction Data



Automated Property Description



Introduction

Today, we'll discuss building an AI-powered recommendation system for TrustAsset, a real estate and vehicle marketplace.



Goal

Our goal is to enhance user experience and engagement by providing personalized recommendations. We're focusing on a content-based approach, which uses item features to find relevant suggestions.

**8**

Dataset

Contains detailed information about residential properties listed or recently sold across the United States, including price, physical characteristics (beds, baths, size, lot size), and location data.

Year	Size	Source	Domain	Dataset Name
2024	Over 2.2 million property listings	Kaggle	Real Estate	USA Real Estate Listings

Design and Train Neural Network for Embeddings



10

Autoencoder Model

- **A type of artificial neural network used for learning efficient data codings in an unsupervised manner. It learns to compress data from the input layer into a short code, and then uncompress that code into an output that is as close as possible to the original input.**
- **Structure: Consists of two main parts: an Encoder and a Decoder.**

**11**

Encoder: Compresses input features

- **Input Layer (Dimension: INPUT_DIM)**
- **Dense(256) -> ReLU, Batch Norm, Dropout(0.3)**
- **Dense(128) -> ReLU, Batch Norm, Dropout(0.3)**
- **Bottleneck (Embedding Layer): Learned representation**
- **Dense(10) -> ReLU activation (Initial Baseline)**
- **Dimension: EMBEDDING_DIM (10)**

**12**

Decoder: Reconstructs original features

- **Dense(128) -> ReLU, Batch Norm, Dropout(0.3)**
- **Dense(256) -> ReLU, Batch Norm, Dropout(0.3)**
- **Output Layer (Dimension: INPUT_DIM) -> Linear activation**



13

Training the Autoencoder

- **Goal: Train the entire Autoencoder network.**
- **Objective: Minimize the difference between the original input data and the reconstructed output data.**
- **Loss Function: Mean Squared Error (MSE) is commonly used.**



14

Key Findings from Experiments

- **Experimentation: Varied embedding dimension, activation, and layer sizes.**
- **Best Performance: Experiment 4 (Wider layers, Linear embedding activation, Dropout 0.2).**
- **Achieved the lowest reconstruction error (MSE/MAE).**
- **Linear activation in embedding layer was beneficial for standardized data.**

A decorative graphic on the left side of the slide. It features a thick green line that starts from the left edge, curves downwards, and then continues horizontally. A thick blue line starts from the bottom edge, curves upwards, and then continues horizontally, overlapping the green line. A thick red line starts from the top edge, curves downwards, and then continues horizontally. An orange circle is positioned to the left of the green line. There are two black dots: one on the green line where it curves, and another on the blue line where it curves.

Autoencoder Model Experiments Comparison

Experiment	Embedding Dim	Embedding Activation	Hidden Layers (Encoder)	Dropout Rate	Best Val MSE	Best Val MAE	Best Epoch	Training Epochs
Baseline	10	'relu'	(256, 128)	0.3	0.0064	0.0431	51	66
Experiment 1	16	'relu'	(256, 128)	0.3	0.0067	0.0588	11	26
Experiment 2	10	'linear'	(256, 128)	0.3	0.0048	0.0408	67	82
Experiment 3	10	'linear'	(512, 256)	0.3	0.0016	0.0236	66	81
Experiment 4 (Best)	10	'linear'	(512, 256)	0.2	0.0009	0.0176	65	80

Generate Item Embeddings



18

Generate Item Embeddings

- **Goal: Obtain the learned numerical representation (embedding) for every property.**
- **Tool: Use the trained Encoder model.**
- **The Encoder is the first half of the Autoencoder we trained.**



19

Process: Passing Data Through the Encoder

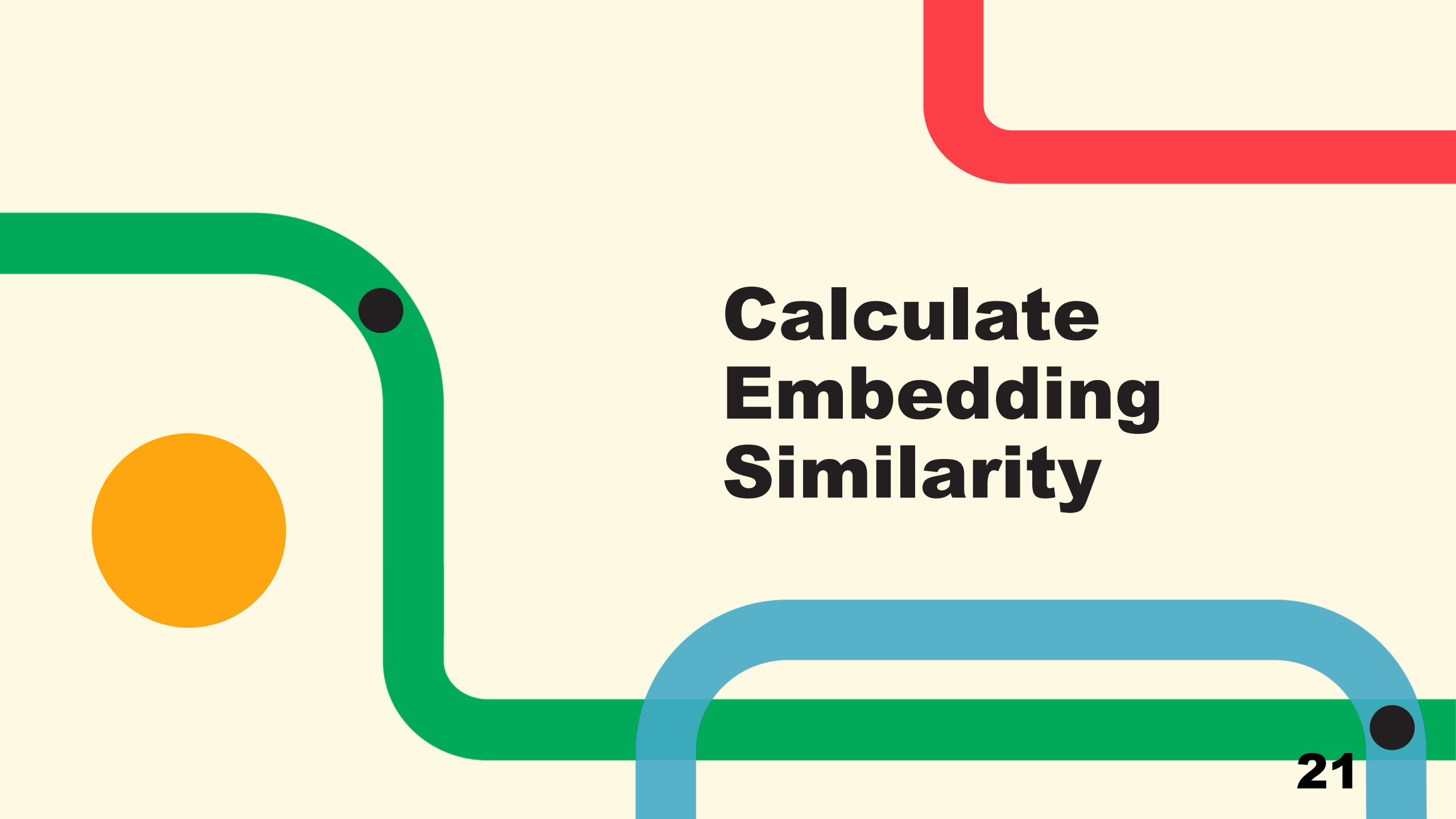
- **Input: The `item_features_matrix` containing the preprocessed features for all 300k properties.**
- **This matrix is fed into the trained Encoder model.**
- **The Encoder applies the learned weights and biases to transform the input features.**



20

Output: The Item Embeddings

- **Result: The output of the Encoder for each property's feature vector is its corresponding item embedding.**
- **This is a dense, lower-dimensional vector (10).**
- **Each embedding vector is a unique "fingerprint" representing the property's content in the learned space.**

An abstract graphic featuring a green line that starts from the left, curves down, and then continues horizontally. A blue line starts from the bottom, curves up, and then continues horizontally, overlapping the green line. A red line starts from the top right and curves down. An orange circle is positioned on the left side of the green line. Two black dots are located on the green line: one on the upper curve and one on the horizontal segment. The text "Calculate Embedding Similarity" is centered in the right half of the image.

Calculate Embedding Similarity

Goal: Find which properties have similar embeddings.

Method: Compute Cosine Similarity between all pairs of embeddings.

Challenge: Full similarity matrix is too large (300k x 300k).

Solution: Chunked calculation – compute similarity in batches and store only the Top N (e.g., 50) most similar pairs per item.

The background features a light cream color with several abstract geometric elements. A thick red line enters from the top right, turns left, and extends horizontally. A thick green line enters from the left, turns down, and then turns right, passing behind a blue line. A thick blue line enters from the bottom, turns left, and then turns down. A solid orange circle is positioned on the left side. Two small black dots are located on the green line: one near the top left and another near the bottom right.

Create Embedding Similarity Lookup

Input: Saved Top N embedding similarity pairs (e.g., in Parquet file).

Structure: Create an in-memory dictionary.

Keys: Source item index.

Values: List of (similar item index, similarity score) tuples.

Purpose: Enables very fast retrieval of similar items for any given property.



Generate Synthetic User Interaction Data



Need: Real user data is unavailable for development.

Solution: Simulate realistic user interactions (views, favorites).

Method: Use LLMs or rule-based logic based on user archetypes and listing features/similarity.

Output: A dataset of simulated user events (who did what, to which item, when).

Alignment: Simulate interactions that match backend schema concepts (Clicks, Favorites, Search Queries, Preferences).



27

Why Simulate Interactions?

Develop & Test: Build and refine the recommendation logic *before* live data integration.

Evaluate Personalization: Assess if the system recommends items similar to a user's *simulated* history.

Mimic Behavior: Create patterns (e.g., browsing similar items) more realistic than random data.

Controlled Environment: Test specific scenarios and user types.

Automated Property Description



29

The Problem: Inefficient & Inconsistent Manual Descriptions

Creating unique, compelling descriptions for many listings is time-consuming. Manual process can lead to inconsistencies in style, tone, and detail. Scalability issues with a large volume of properties. Need for a more efficient and consistent approach.



30

Initial Approach: Using a Pre-trained LLM

- **Goal: Generate descriptions automatically from structured data.**
- **Model: Started with deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B (a general-purpose LLM).**
- **Process:**
 - **Load structured property data (e.g., price, beds, baths, location).**
 - **Format data into a text "Prompt" for the LLM.**
 - **LLM generates a description based on its general training.**
- **Outcome: Successfully generated descriptions, but the style was generic.**



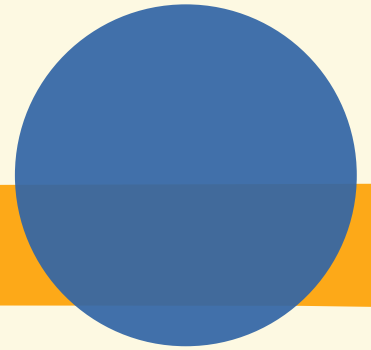
31

Refining Output: The Need for Fine-tuning

**Pre-trained models are generalists.
Real estate descriptions require a
specific style, tone, and focus.**

**Fine-tuning: Adapting a pre-trained
model to a specific task/domain
using a smaller, relevant dataset.**

**Objective: Teach the LLM to
generate descriptions in the desired
"Trustasset style."**





32

Fine-tuning Experiment: Adapting the Model

- **Dataset: Small (10 properties), manually crafted (Prompt, Desired Description).**
- **Tools: Hugging Face transformers (Trainer API).**
- **Process: Load model, tokenize data, configure, train.**



**33**

Training Challenges: Memory Constraints

- **LLM fine-tuning is memory-intensive.**
- **OutOfMemoryError encountered.**
- **1.5B parameter model was too large for direct training on available GPU.**



34

Enabling Training with Limited Memory

- **Solutions: Gradient Accumulation:** Simulate larger batches with less memory.
- **Quantization (BitsAndBytes):** Load model in lower precision (4-bit) to reduce size.
- **LoRA (Low-Rank Adaptation):** Train only small adapter layers for efficiency.

Pre-trained vs. Fine-tuned Model Comparison (Property Description Generation)

Aspect	Base Pre-trained Model (deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B)	Fine-tuned Model (with LoRA adapters)
Training Data	Massive general internet text	Massive general internet text + Small (10 examples) task-specific data
Model Parameters	All 1.5 Billion parameters	All 1.5 Billion parameters (base frozen) + Small LoRA adapter parameters (trainable)
Training Process	Extensive pre-training by model developers	Additional training on a small dataset using PEFT (LoRA)
Memory Requirement	High (for loading and inference)	Lower (for training/loading due to LoRA & quantization)
Adaptation	None (general purpose)	Adapted to the specific task/style of the fine-tuning data
Observed Output (from comparison script)	Generic, sometimes repetitive, lacks specific real estate style.	Unexpectedly poor, repetitive, sometimes nonsensical output.
Reason for Observed Output	Trained on general text, no specific real estate domain knowledge.	Severe overfitting and lack of generalization due to extremely limited fine-tuning data (only 10 examples).

The background features abstract, thick, rounded lines in red and orange. On the left, a vertical orange line runs down the page, with two horizontal red lines crossing it. A small black dot is located at the intersection of the orange line and the lower red line. On the right, a vertical red line runs down the page, with a small black dot near the top and a large green circle near the bottom. The text "Thank you" is centered in the middle of the page.

**Thank
you**