HUIT.Meow

Vietnam Datathon 2023

Members

- Do The Sang (leader)
- Nguyen Truong Phat
- Truong Tran Bao Long

- From Ho Chi Minh City University of Industry and Trade (HUIT)

Contents

1. Introduction

Sales and Inventory Data of Vietnam Retailers

1. Introduction

Dataset: we use Dataset 2 – Sales and Inventory Data of Vietnam Retailers

1. Introduction (cont'd)

Overview of Problem:

- The company is collecting sales data on a monthly basis from 2022 to 2023. Each month's data is stored in a file with a specific naming convention, indicating the month and year. The data includes various fields such as month, week, site, branch_id, channel_id, distribution_channel, distribution_channel_code, sold_quantity, cost_price, net_price, customer_id, and product_id.
- The challenge is to effectively analyze and derive insights from this sales data. This involves understanding sales patterns, identifying top-performing products, optimizing distribution channels, and managing costs. Additionally, the company may want to track customer behavior and preferences to improve marketing and sales strategies.

1. Introduction (cont’d)

Purpose:

In this context, the Minimum Viable Product (MVP) is conceived as a fundamental system or tool designed to meet the company's immediate requirements for sales data analysis. The aim of the MVP is to deliver a practical solution with essential features, enabling the company to swiftly derive value from the data. Acting as a cornerstone, the MVP is intended to pave the way for subsequent development and improvement.

It's worth noting that we employ three distinct methods to address the long tail problem: Content-Based Filtering, Collaborative Filtering, and DNN (Deep Neural Network).

1. Introduction (cont'd)

Components of the MVP:

1. Data Ingestion and Processing:

- Develop a script or tool to automatically ingest data from the monthly files into a centralized database or data warehouse.
- Implement a process to clean and preprocess the data, handling any missing values or outliers.

1. Introduction (cont'd)

Components of the MVP:

2. Basic Dashboard or Reports:

- Create a simple dashboard or set of reports to visualize key metrics such as total sales, sold quantity, and costs over time.
- Include visualizations for top-performing products, distribution channels, and branches.

1. Introduction (cont'd)

Components of the MVP:

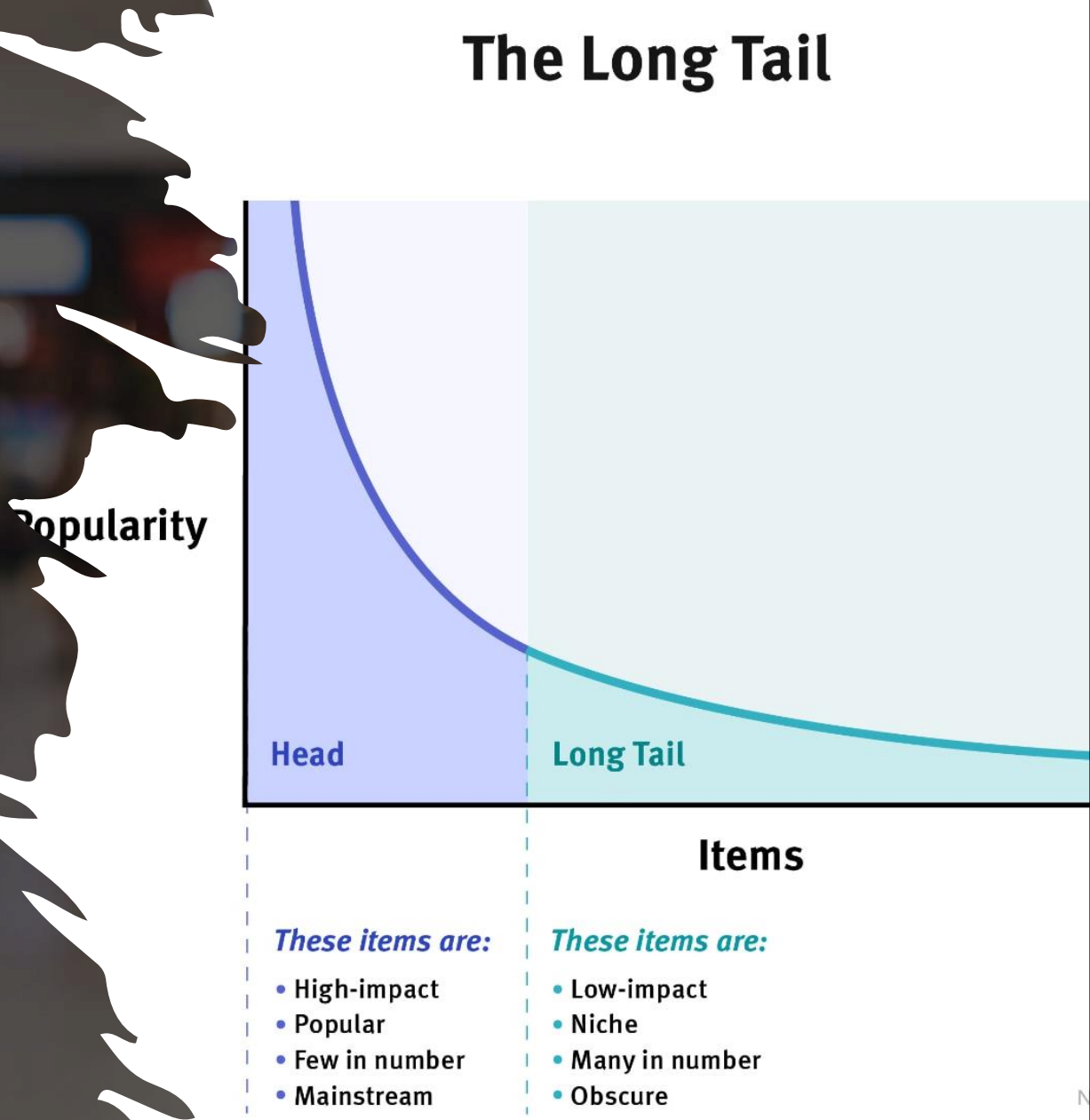
3. Basic Analysis Features:

- Implement basic analysis functions to calculate metrics like profit margins, average selling prices, and sales growth.
- Include filters and parameters to allow users to focus on specific time periods, products, or channels.

2. Problem Statement

2. Problem Statement

The challenge in our dataset is the **Long Tail** phenomenon in transactions, where high-margin items are often left in warehouses, affecting sales and product quality. To solve this problem, the team proposed to implement a Recommendation System algorithm. This is important because customers may miss out on desirable items in the store, leading to accessibility issues and potential loss of sales. Due to space constraints, the store can only display a limited number of products, with a significant portion of sales coming from a small portion of popular items. This imbalance, known as the long tail phenomenon, highlights the need to optimize product visibility and sales strategies for both popular and less popular products.



2. Problem Statement

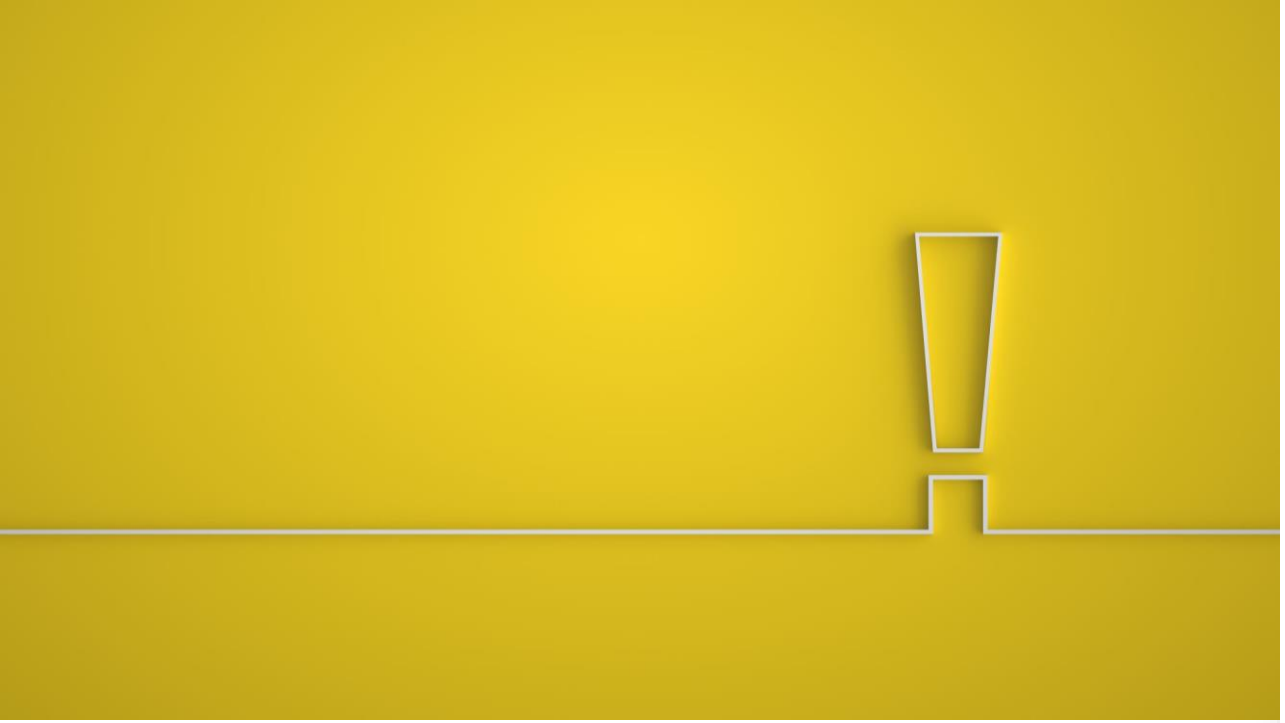
- Paint Points of Long Tail
  - The traditional approach to retail, particularly in the fashion industry, presents several pain points and inefficiencies. Physical stores face limitations in display space, leading to the prioritization of a small set of popular products while relegating less-known items to the background. This not only hinders the discovery of niche or unique fashion pieces by customers but also results in missed sales opportunities for these less-prominent products.


3. Solution Overview

3. Solution Overview

Long Tail is a problem in **Recommender System**. Recommender System can be divided into 2 groups:

1. *Content-Based* systems focus on properties of items. Similarity of items is determined by measuring the similarity in their properties.
2. *Collaborative-Filtering* systems focus on the relationship between users and items. Similarity of items is determined by the similarity of the ratings of those items by the users who have rated both items.



	item1	item2	item3	item4
user1	2	5	1	3
user2	4	?	?	1
user3	?	4	2	?
user4	2	4	3	1
user5	1	3	2	?

3. Solution Overview - Collaborative Filtering

Build Utility Matrix

If we don't have Utility Matrix, we can't recommend products to users.

The matrix in which:

- Each row represents a user
- Each column represents an item (fashion product)

To simplify, we will assume that the feedback matrix is binary; that is, a value of 1 indicates interest in or buy the product.

3. Solution Overview - Collaborative Filtering (cont’d)

There is a method in Collaborative Filtering called Neighborhood-based Collaborative. The two most crucial questions in a Neighborhood-based Collaborative Filtering system are:

1. How to determine the similarity between two users?
2. Once similar users are identified, how to predict a user's level of interest in an item?

The process of determining a user's interest in an item based on the interest levels of similar users is known as User-user collaborative filtering. Another approach considered more effective is Item-item collaborative filtering. In this approach, instead of identifying user similarities, the system focuses on determining similarities between items. Consequently, the system suggests items similar to those in which the user has a high level of interest.

3. Solution Overview - Content-based Filtering

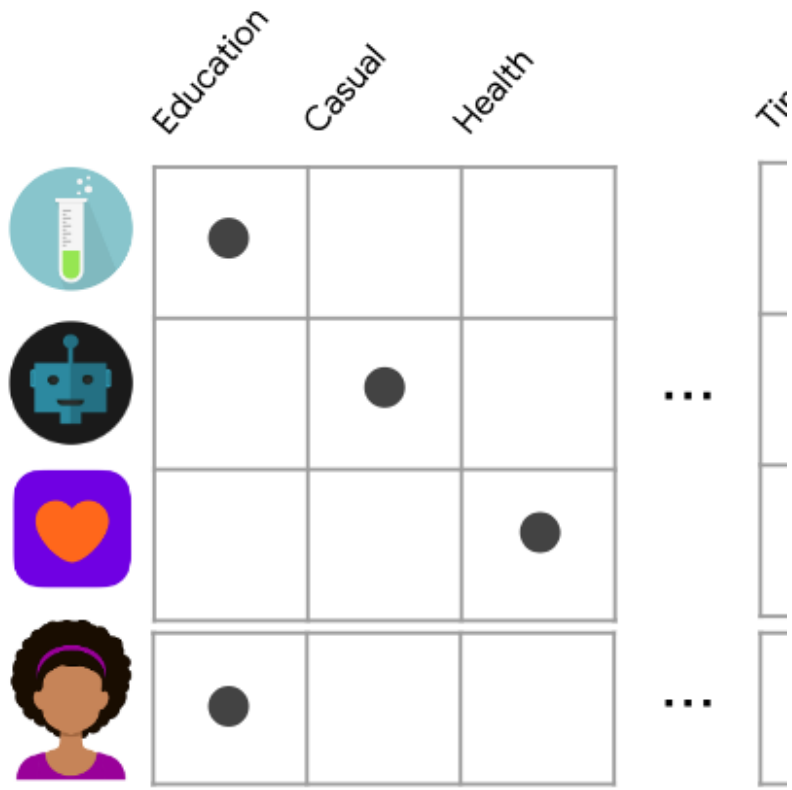
- Content-based filtering uses item features to recommend other items similar to what the user likes, based on their previous actions or explicit feedback.

2. Solution Overview - Content-based Filtering (cont'd)

Using Dot Product as a Similarity Measure

Consider the case where the user embedding x and y the app embedding are both binary vectors. Since $\langle x, y \rangle = \sum_{i=1}^{d} x_i y_i$, a feature appearing in both x and y contributes a 1 to the sum. In other words, $\langle x, y \rangle$ is the number of features that are active in both vectors simultaneously. A high dot product then indicates more common features, thus a higher similarity.

So, Which app should we recommend? → The educational app created by Science R Us. This item has the highest dot product at 2. Our user really likes science and educational apps.



2. Solution Overview - Content-based Filtering Advantages & Disadvantages

- **Advantages**
- The model doesn't need any data about other users, since the recommendations are specific to this user. This makes it easier to scale to a large number of users.
- The model can capture the specific interests of a user, and can recommend niche items that very few other users are interested in.
- **Disadvantages**
- Since the feature representation of the items are hand-engineered to some extent, this technique requires a lot of domain knowledge. Therefore, the model can only be as good as the hand-engineered features.
- The model can only make recommendations based on existing interests of the user. In other words, the model has limited ability to expand on the users' existing interests.

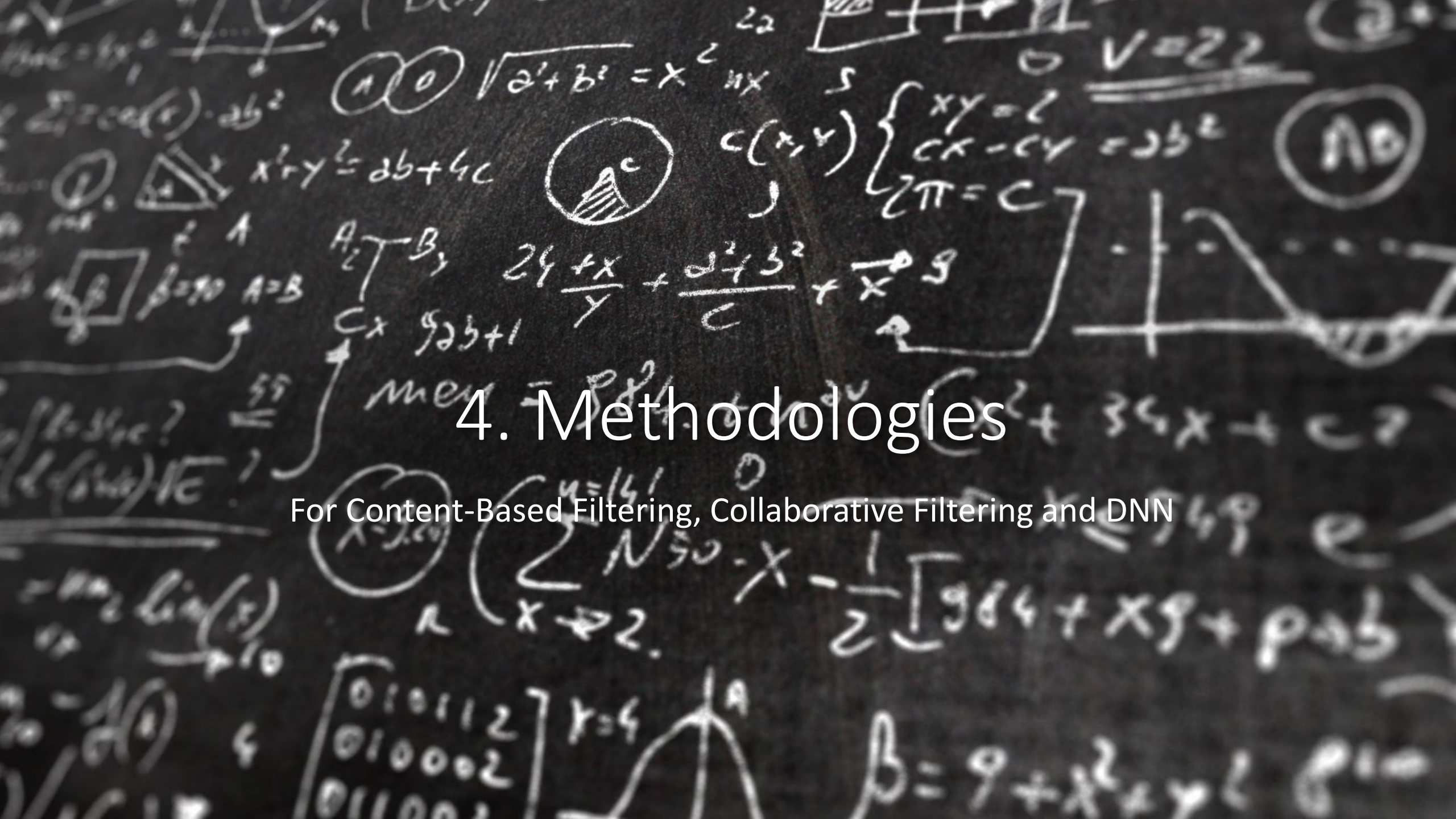
3. Solution Overview - Deep Neural Network Models

- Deep neural network (DNN) models can address these limitations of matrix factorization. DNNs can easily incorporate query features and item features (due to the flexibility of the input layer of the network), which can help capture the specific interests of a user and improve the relevance of recommendations.
- One possible DNN model is Softmax.

3. Solution Overview - Softmax DNN for Recommendation

Softmax, which treats the problem as a multiclass prediction problem in which:

- The input is the user query.
- The output is a probability vector with size equal to the number of items in the corpus, representing the probability to interact with each item; for example, the probability to click on or watch a YouTube video.



4. Methodologies

For Content-Based Filtering, Collaborative Filtering and DNN

4.1. Content-Based Recommendations

	A	B	C	D	E	F	item's feature vectors
Adidas Shoes	5	5	0	0	1	?	$x_1 = [0.99, 0.02]$
Nike Shoes	5	?	?	0	?	?	$x_2 = [0.91, 0.11]$
Adidas Shirt	?	4	1	?	?	1	$x_3 = [0.95, 0.05]$
Nike Shirt	1	1	4	4	4	?	$x_4 = [0.01, 0.99]$
Bitits Shoes	1	0	5	?	?	?	$x_5 = [0.03, 0.98]$
User's models	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	← Need to optimize

4.1. Content-Based Recommendations (cont'd)

Problem Definition: Finding User Models

- Task: Search for a model θ_i for each user
- Regression Problem: When ratings form a continuous range of values.
- Classification Problem: When ratings fall into specific categories (e.g., like/dislike).
- Training Data: Comprises pairs (item profile, ratings) for items a user has rated.
- Objective: Construct a model θ_i based on training data.

Matrix Utility and Missing Values

- Matrix Utility: Represents user-item interactions.
- Missing Values: Represent unrated items.
- Prediction: Filling in missing values by applying θ_i to unrated items.

Model Selection: Regression vs. Classification

- Choice depends on the application.
- Example: Linear Regression with regularization, specifically Ridge Regression.

4.1. Content-Based Recommendations (cont'd)

Loss Function

Assume that we can find a model for each user, represented by a column vector of coefficients $\boldsymbol{w}_i$ and bias b_n, so that the level of interest of a user in an item can be calculated by a linear function:

$$y_{mn} = \boldsymbol{x}_m \boldsymbol{w}_n + b_n$$

Note that x_m is row vector and w_n is column vector.

4.1. Content-Based Recommendations (cont'd)

Loss Function (cont'd)

For any arbitrary user n, considering the training set as a collection of filled-in components of y_n, we can construct a loss function similar to Ridge Regression as follows:

$$\mathcal{L}_n = \frac{1}{2} \sum_{m:r_{mn}=1} (\boldsymbol{x}_m \boldsymbol{w}_n + b_n - y_{mn})^2 + \frac{\lambda}{2} \|\boldsymbol{w}_n\|_2^2$$

In this equation, the second component is the regularization term, where λ is a positive parameter. Note that regularization is typically not applied to the bias b_n.

4.1. Content-Based Recommendations (cont'd)

Loss Function (cont'd)

In practice, the mean squared error is often used, and the loss $\mathcal{L}_n$ is rewritten as:

$$\mathcal{L}_n = \frac{1}{2S_n} \sum_{m:r_{mn}=1} (\boldsymbol{x}_m \boldsymbol{w}_n + b_n - y_{mn})^2 + \frac{\lambda}{2S_n} \|\boldsymbol{w}_n\|_2^2$$

Where S_n is the number of items that user n has rated.

4.1. Content-Based Recommendations (cont'd)

Loss Function (cont'd)

Since the loss function expression depends only on the items that have been rated, we can simplify it by defining $\widehat{\boldsymbol{y}}_{\mathbf{n}}$ as a sub-vector of $\boldsymbol{y}$ constructed by extracting the components with non-missing values in column n, meaning those rated by user n in the Utility Matrix $\boldsymbol{Y}$. Additionally, let $\widehat{\boldsymbol{X}}_{\mathbf{n}}$ be a sub-matrix of the feature matrix $\boldsymbol{X}$, created by extracting the rows corresponding to items rated by user n. (Refer to the example below for a clearer understanding). In this case, the model's loss function expression for user n can be written more concisely as:

$$\mathcal{L}_n = \frac{1}{2S_n} \sum_{m:r_{mn}=1} \left(\widehat{\boldsymbol{X}}_m \boldsymbol{w}_n + b_n \boldsymbol{e}_n - \boldsymbol{y}_{mn}\right)^2 + \frac{\lambda}{2S_n} \|\boldsymbol{w}_n\|_2^2$$

where $\boldsymbol{e}_n$ is a column vector containing S_n elements of 1.

4.1. Content-Based Recommendations (cont'd)

Loss Function (cont'd)

Returning to the example in previous table, the feature matrix for the items (each row corresponding to an item) is:

$$\boldsymbol{X} = \begin{bmatrix} .99 & .02 \\ .91 & .11 \\ .95 & .05 \\ .01 & .99 \\ .03 & .98 \end{bmatrix}$$

Considering the case of user E with $n = 5, \boldsymbol{y}_5 = [1, ?, ?, 4, ?]^T \Rightarrow \boldsymbol{r}_5 = [1,0,0,1,0]^T$. Since E has only rated items in the first and fourth positions, $s_5 = 2$. Furthermore:

$$\widehat{\boldsymbol{X}}_5 = \begin{bmatrix} .99 & .02 \\ .01 & .99 \end{bmatrix}, \boldsymbol{y}_5 = \begin{bmatrix} 1 \\ 4 \end{bmatrix}, \boldsymbol{e}_5 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

In that case, the loss function for the coefficient corresponding to user E is:

$$\mathcal{L}_n = \frac{1}{4} \left\| \begin{bmatrix} .99 & .02 \\ .01 & .99 \end{bmatrix} \boldsymbol{w}_5 + b_5 \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 4 \end{bmatrix} \right\|_2^2 + \frac{\lambda}{4} \|\boldsymbol{w}_5\|_2^2$$

4.2. Collaborative Filtering

***User-user* Collaborative Filtering**

Similarity functions
$sim(u_0, u_1) > sim(u_0, u_i), \forall i > 1$

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	5	5	2	0	1	?	?
i_1	3	?	?	0	?	?	?
i_2	?	4	1	?	?	1	2
i_3	2	2	3	4	4	?	4
i_4	2	0	4	?	?	?	5

4.2. Collaborative Filtering

User-user Collaborative Filtering (cont'd)

Example illustrating User-User Collaborative Filtering.

a) Original Utility Matrix.

b) Normalized Utility Matrix.

c) User Similarity Matrix.

d) Prediction of missing (normalized) ratings. e) Example of predicting the normalized rating of u_1 for i_1.

f) Prediction of missing (denormalized) ratings.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	5	5	2	0	1	?	?
i_1	4	?	?	0	?	2	?
i_2	?	4	1	?	?	1	1
i_3	2	2	3	4	4	?	4
i_4	2	0	4	?	?	?	5
	↓	↓	↓	↓	↓	↓	↓
$\bar{u}_j$	3.25	2.75	2.5	1.33	2.5	1.5	3.33

a) Original utility matrix $\mathbf{Y}$ and mean user ratings.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	1.75	2.25	-0.5	-1.33	-1.5	0	0
i_1	0.75	0	0	-1.33	0	0.5	0
i_2	0	1.25	-1.5	0	0	-0.5	-2.33
i_3	-1.25	-0.75	0.5	2.67	1.5	0	0.67
i_4	-1.25	-2.75	1.5	0	0	0	1.67

b) Normalized utility matrix $\bar{\mathbf{Y}}$.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
u_0	1	0.83	-0.58	-0.79	-0.82	0.2	-0.38
u_1	0.83	1	-0.87	-0.40	-0.55	-0.23	-0.71
u_2	-0.58	-0.87	1	0.27	0.32	0.47	0.96
u_3	-0.79	-0.40	0.27	1	0.87	-0.29	0.18
u_4	-0.82	-0.55	0.32	0.87	1	0	0.16
u_5	0.2	-0.23	0.47	-0.29	0	1	0.56
u_6	-0.38	-0.71	0.96	0.18	0.16	0.56	1

c) User similarity matrix $\mathbf{S}$.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	1.75	2.25	-0.5	-1.33	-1.5	0.18	-0.63
i_1	0.75	0.48	-0.17	-1.33	-1.33	0.5	0.05
i_2	0.91	1.25	-1.5	-1.84	-1.78	-0.5	-2.33
i_3	-1.25	-0.75	0.5	2.67	1.5	0.59	0.67
i_4	-1.25	-2.75	1.5	1.57	1.56	1.59	1.67

d) $\hat{\mathbf{Y}}$

Predict normalized rating of u_1 on i_1 with $k = 2$

Users who rated i_1 : $\{u_0, u_3, u_5\}$

Corresponding similarities: {0.83, -0.40, -0.23}

$\Rightarrow$ most similar users: $\mathcal{N}(u_1, i_1) = \{u_0, u_5\}$

with **normalized ratings** {**0.75, 0.5**}

$$\Rightarrow \hat{y}_{i_1,u_1} = \frac{0.83*0.75+(-0.23)*0.5}{0.83+|-0.23|} \approx 0.48$$

e) Example

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	5	5	2	0	1	1.68	2.70
i_1	4	3.23	2.33	0	1.67	2	3.38
i_2	4.15	4	1	-0.5	0.71	1	1
i_3	2	2	3	4	4	2.10	4
i_4	2	0	4	2.9	4.06	3.10	5

f) Full $\mathbf{Y}$

4.2. Collaborative Filtering

User-user Collaborative Filtering (cont'd)

Cosine Similarity

$$cosinesim(\boldsymbol{u}_1, \boldsymbol{u}_2) = \cos(\boldsymbol{u}_1, \boldsymbol{u}_2) = \frac{\boldsymbol{u}_1^T \boldsymbol{u}_2}{\|\boldsymbol{u_1}\|_2 \cdot \|\boldsymbol{u}_2\|_2}$$

4.2. Collaborative Filtering

Item-item Collaborative Filtering

Example illustrating Item-Item Collaborative Filtering.

a) Original Utility Matrix.

b) Normalized Utility Matrix.

c) User Similarity Matrix.

d) Prediction of missing (normalized) ratings.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6		
i_0	5	5	2	0	1	?	?	→	2.6
i_1	4	?	?	0	?	2	?	→	2
i_2	?	4	1	?	?	1	1	→	1.75
i_3	2	2	3	4	4	?	4	→	3.17
i_4	2	0	4	?	?	?	5	→	2.75

a) Original utility matrix $\mathbf{Y}$ and mean item ratings.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	2.4	2.4	-.6	-2.6	-1.6	0	0
i_1	2	0	0	-2	0	0	0
i_2	0	2.25	-0.75	0	0	-0.75	-0.75
i_3	-1.17	-1.17	-0.17	0.83	0.83	0	0.83
i_4	-0.75	-2.75	1.25	0	0	0	2.25

b) Normalized utility matrix $\bar{\mathbf{Y}}$.

	i_0	i_1	i_2	i_3	i_4
i_0	1	0.77	0.49	-0.89	-0.52
i_1	0.77	1	0	-0.64	-0.14
i_2	0.49	0	1	-0.55	-0.88
i_3	-0.89	-0.64	-0.55	1	0.68
i_4	-0.52	-0.14	-0.88	0.68	1

c) Item similarity matrix $\mathbf{S}$.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	2.4	2.4	-.6	-2.6	-1.6	-0.29	-1.52
i_1	2	2.4	-0.6	-2	-1.25	0	-2.25
i_2	2.4	2.25	-0.75	-2.6	-1.20	-0.75	-0.75
i_3	-1.17	-1.17	-0.17	0.83	0.83	0.34	0.83
i_4	-0.75	-2.75	1.25	1.03	1.16	0.65	2.25

d) Normalized utility matrix $\bar{\mathbf{Y}}$.

4.3. Recommendation Using Deep Neural Network Models

Deep neural network (DNN) models can address these limitations of matrix factorization. DNNs can easily incorporate query features and item features (due to the flexibility of the input layer of the network), which can help capture the specific interests of a user and improve the relevance of recommendations.

One possible DNN model is softmax, which treats the problem as a multiclass prediction problem in which:

- The input is the user query.
- The output is a probability vector with size equal to the number of items in the corpus, representing the probability to interact with each item; for example, the probability to click on or watch a YouTube video.

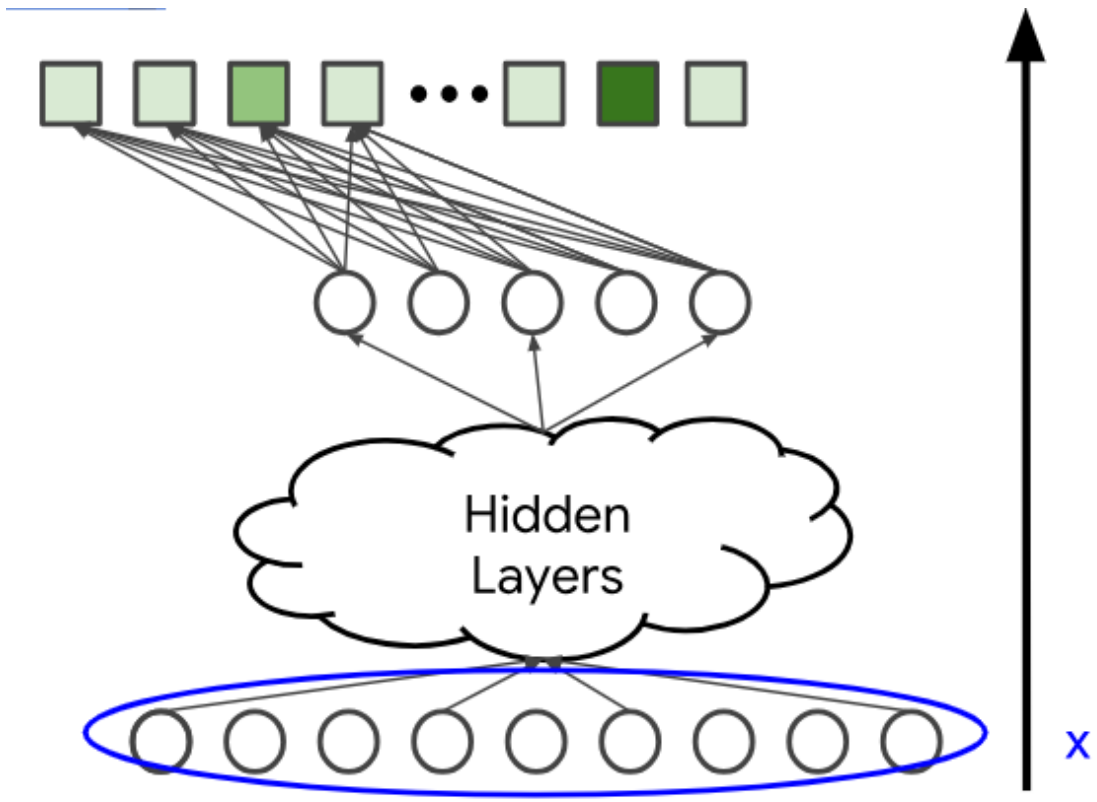
4.3. Recommendation Using Deep Neural Network Models (cont'd)

Input Layer

The input to a DNN can include:

- dense features (for example, **cost_price** and **net_price**)
- sparse features (for example, **month** and **channel_id**)

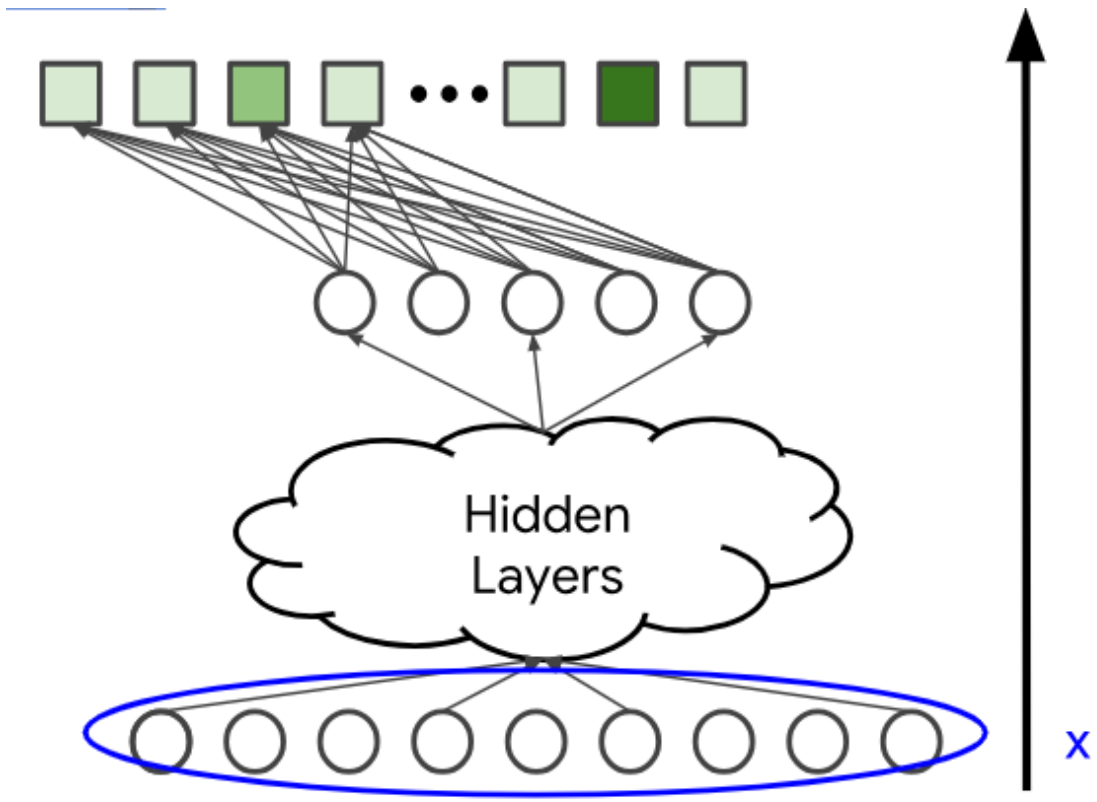
Unlike the matrix factorization approach, you can add side features such as age or country. We'll denote the input vector by x.



4.3. Recommendation Using Deep Neural Network Models (cont'd)

Model Architecture

The model architecture determines the complexity and expressivity of the model. By adding hidden layers and non-linear activation functions (for example, ReLU), the model can capture more complex relationships in the data. However, increasing the number of parameters also typically makes the model harder to train and more expensive to serve. We will denote the output of the last hidden layer by $\psi(x) \in \mathbb{R}^d$.



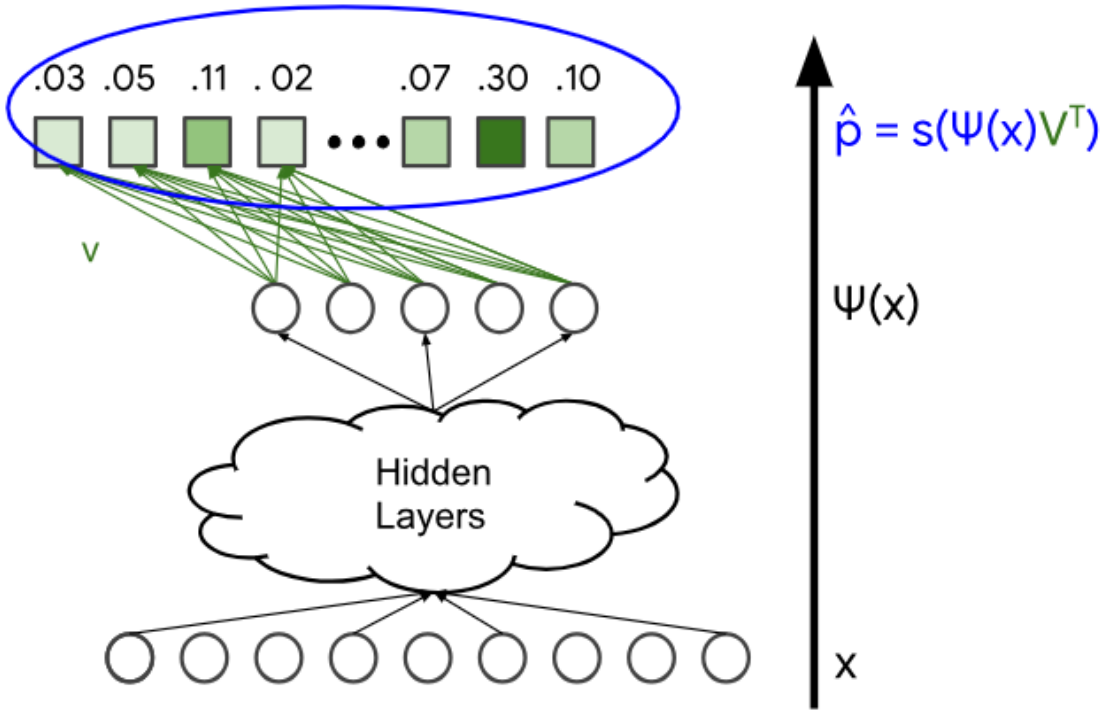
4.3. Recommendation Using Deep Neural Network Models (cont'd)

Softmax Output: Predicted Probability Distribution

The model map the output of the last layer, $\psi(x)$, through a softmax layer to probability distribution $\hat{p} = h(\psi(x)V^T)$

- $h: \mathbb{R}^n \to \mathbb{R}^n$ is the softmax function, given by $h(y)_i = \frac{e^{y_i}}{\sum_j e^{y_i}}$
- $V \in \mathbb{R}^{n \times d}$ is the matrix of weight of the softmax layer.

The softmax layer maps a vector of scores (sometimes called the logits) to a probability distribution.



4.3. Recommendation Using Deep Neural Network Models (cont'd)

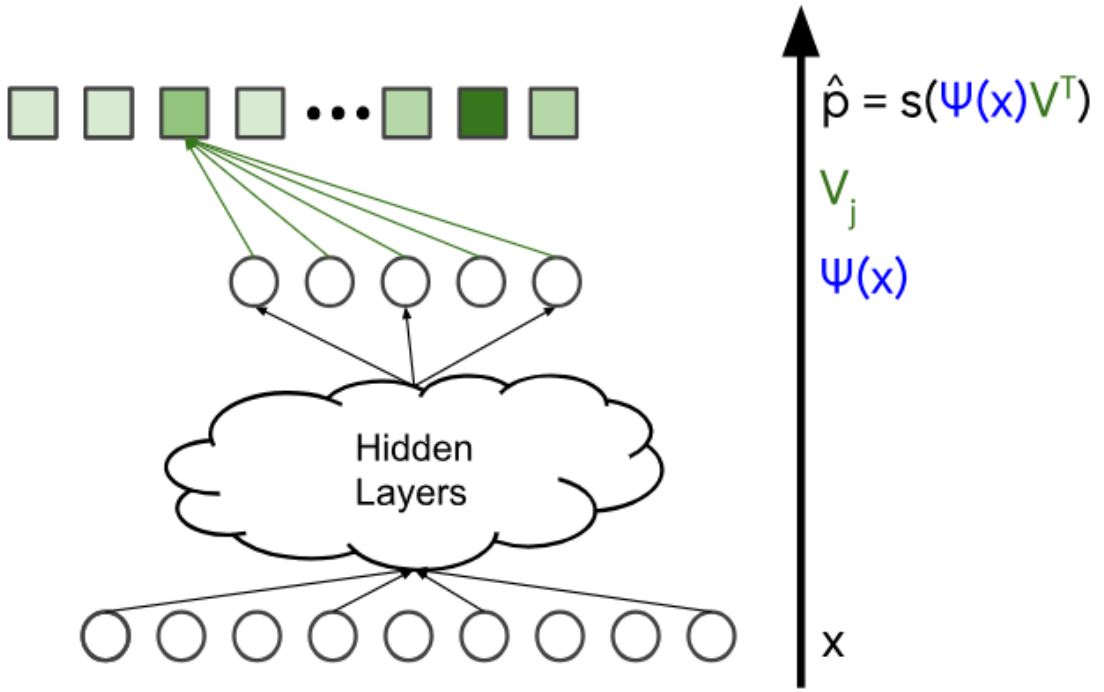
Softmax Embeddings

The probability of item j is given by $\hat{p}_j = \frac{\exp(\langle \psi(x), V_j \rangle)}{Z}$, where Z is a normalization constant that does not depend on j.

In other words, $\log(\hat{p}_j) = \langle \psi(x), V_j \rangle - \log(Z)$, so the log probability of an item j is (up to an additive constant) the dot product of two d-dimensional vectors, which can be interpreted as query and item embeddings:

$\psi(x) \in \mathbb{R}^d$ is the output of the last hidden layer. We call it the embedding of the query x.

$V_j \in \mathbb{R}^d$ is the vector of weights connecting the last hidden layer to output j. We call it the embedding of item j.



5. Core Functionallity

5. Core Functionality (cont’d)

Our MVP product is a tool designed to recommend requested items based on a customer's purchasing history. The tool not only identifies and recommends highly sought-after products, but also includes essential items that closely match the user's needs. To solve the previously mentioned problem, the proposed tool addresses the Long Tail phenomenon in transactions by leveraging customer purchase data. It enhances user experience by ensuring that not only popular items are recommended but also essential products according to personal preferences. This recommendation system optimizes warehouse space utilization and improves overall revenue and product quality. Thereby helping businesses improve revenue

5. Core Functionality (cont’d)

- **Collect Purchase History**

Integrate functionality to collect and analyze customer purchase history to better understand their personal preferences and needs.

- **Personal Recommendation System**

From purchasing history data, develop a product recommendation system that customers prefer

- **Product Classification and Prioritization**

Automatically classify products based on popularity and consumption, helping businesses decide the location and display quantity of each product.

5. Core Functionality (cont’d)

- **Notifications and Cross-Marketing**

Provide intelligent product notifications or cross-marketing to inform about new or less popular products, guiding customers to explore more options. From there, propose to advertise many products to customers

- **Track and Evaluate Performance**

Integrate tools to track and measure the performance of product recommendations, helping businesses better understand how customers interact with recommendations. Simultaneously, offer precise data on revenue generation, identify existing challenges, and propose strategies to overcome these hurdles.

5. Why do businesses need our products?

Optimized Recommendations

Our tool utilizes advanced recommendation algorithms based on customers' purchase history. This ensures that businesses can offer personalized suggestions to their customers, improving the chances of successful sales.

The recommendation tool enhances the overall customer experience by providing tailored suggestions. Customers are more likely to find and purchase items they are interested in, leading to increased satisfaction and loyalty.

5. Why do businesses need our products? (cont'd)

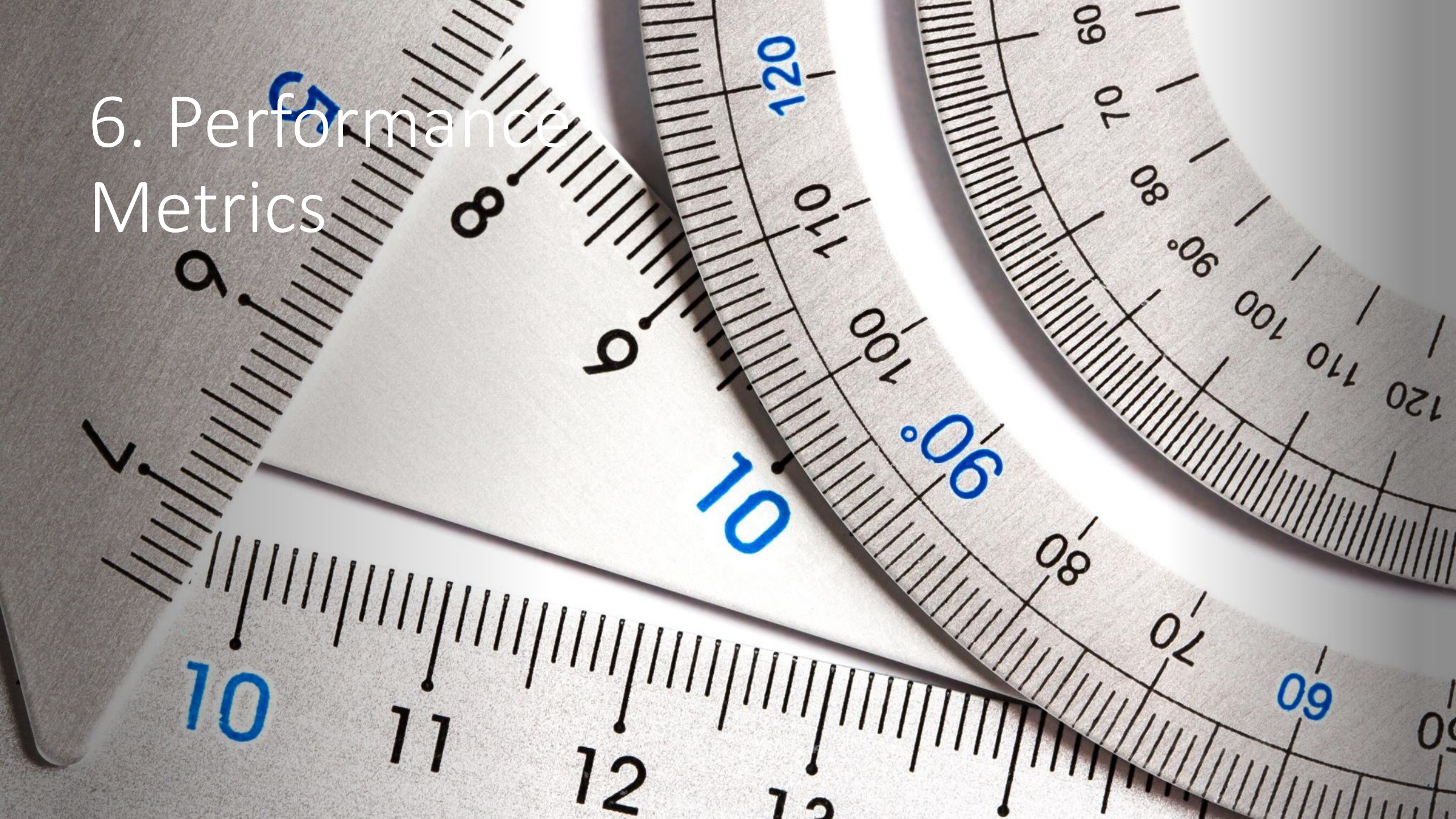
Optimal Use of Warehouse Space

Our tool assists in optimizing the use of this space by ensuring that items with varying levels of popularity are strategically placed and recommended, preventing backlogs and improving product availability.

Increased Revenue

The personalized recommendations and effective management of product visibility contribute to increased sales. By addressing the Long Tail phenomenon and catering to diverse customer preferences, businesses can tap into previously untapped revenue sources.

6. Performance Metrics



6.1. Performance Metrics for Content-based Filtering

- Loss function:

$$\mathcal{L}_n = \frac{1}{2S_n} \sum_{m:r_{mn}=1} \left(\widehat{\boldsymbol{X}}_m \boldsymbol{w}_n + b_n \boldsymbol{e}_n - \boldsymbol{y}_{mn}\right)^2 + \frac{\lambda}{2S_n} \|\boldsymbol{w}_n\|_2^2$$

- RMSE: To assess the model's accuracy, we use Root Mean Squared Error (RMSE). This metric measures the average difference between predicted and actual ratings, with lower values indicating better performance. We compute RMSE for both training and test datasets by comparing predicted and true ratings, providing insights into the model's generalization to new data.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \|y(i) - \hat{y}(i)\|^2}{N}}$$

6.1. Performance Metrics for Collaborative Filtering

- RMSE: For this method, we also use RMSE to evaluate the model performance.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \|y(i) - \hat{y}(i)\|^2}{N}}$$

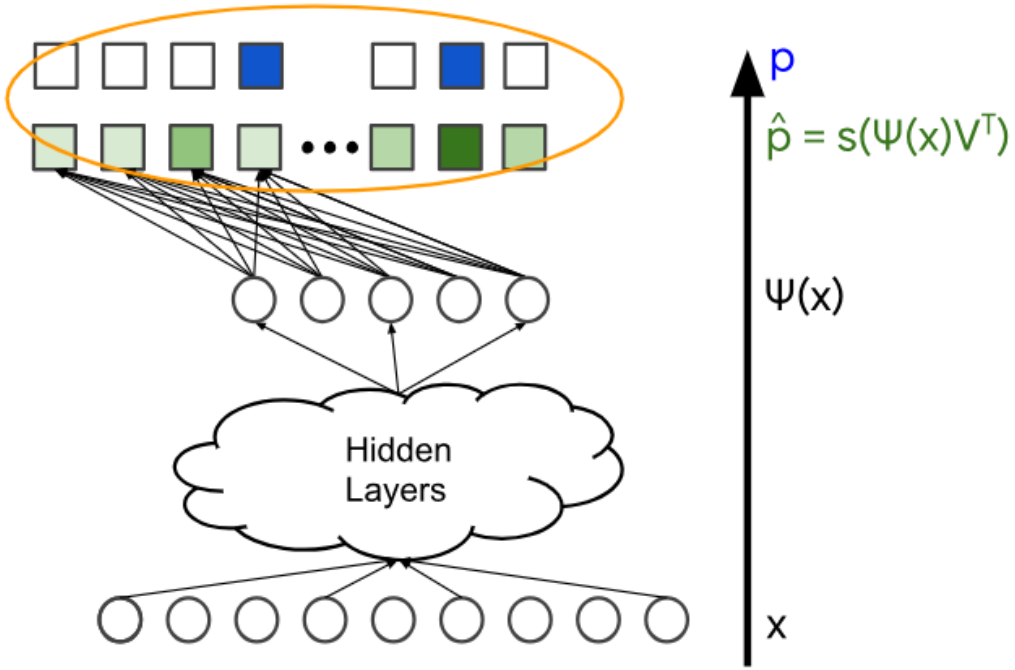
6.1. Performance Metrics for DNN

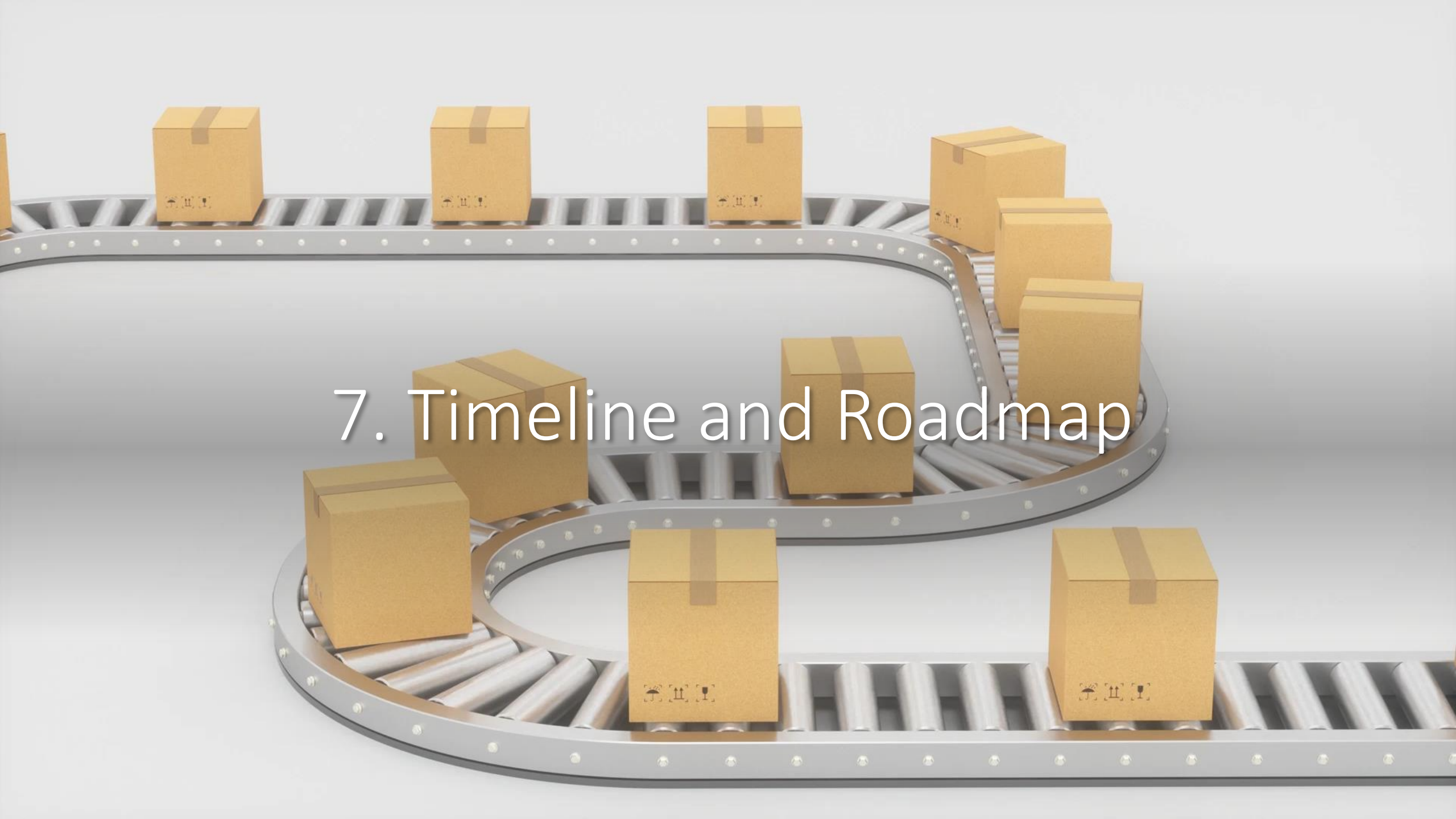
Loss Function

Finally, define a loss function that compares the following:

- $\hat{p}$, the output of the softmax layer (a probability distribution)
- p, the ground truth, representing the items the user has interacted with.This can be represented as a normalized multi-hot distribution (a probability vector).

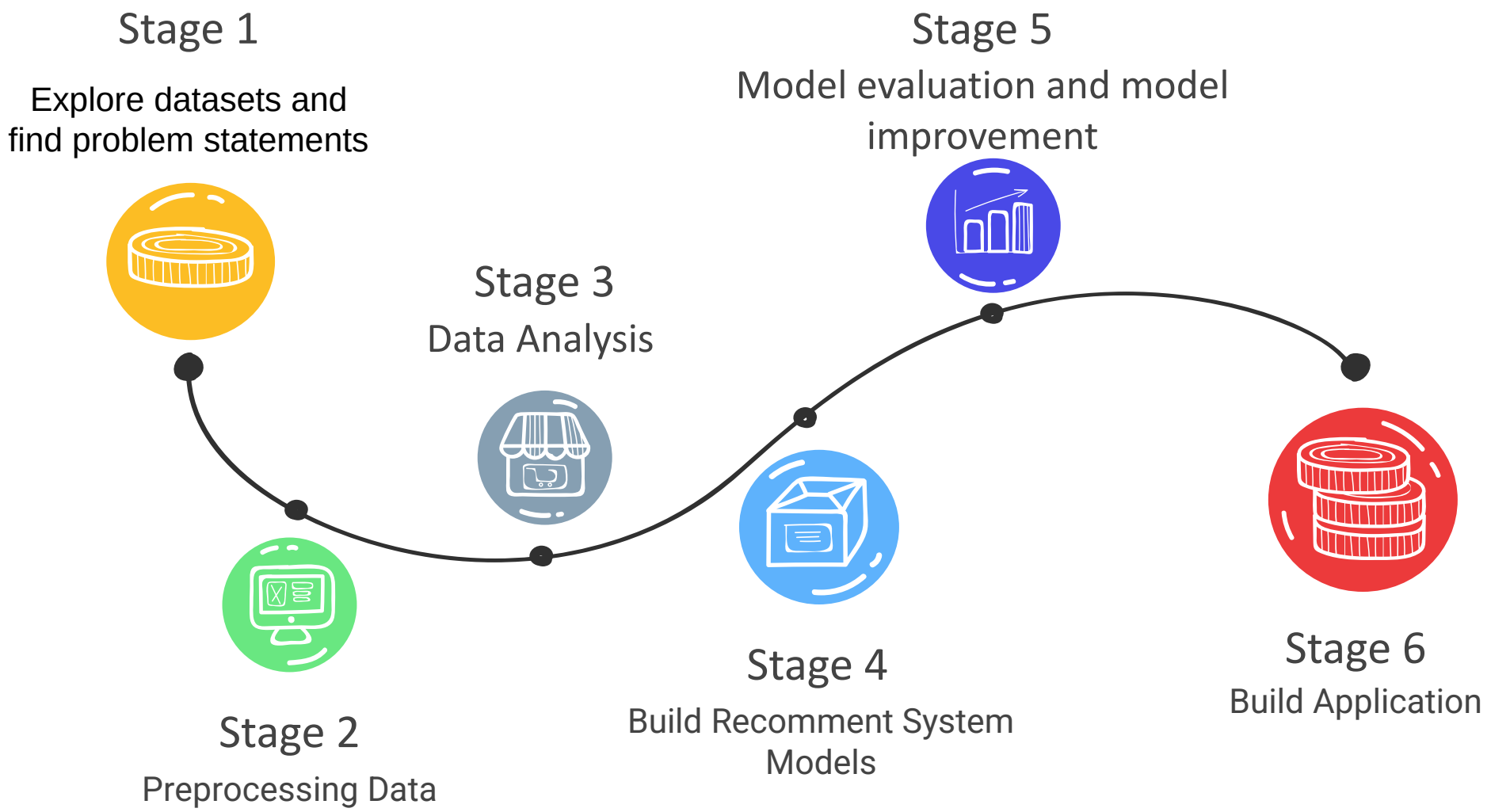
For example, you can use the cross-entropy loss since you are comparing two probability distributions.





7. Timeline and Roadmap

7. Timeline Infographics



8. Conclusion

Summarize the key points

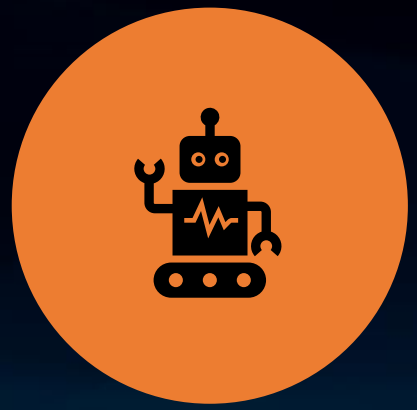
- Long Tail Challenge: Our MVP acknowledges and tackles the Long Tail phenomenon, ensuring that both popular and less popular items are strategically recommended to optimize revenue streams.

- AI model : Our MVP utilizes state-of-the-art Recommendation System Models to empower the tool in analyzing and comprehending individual customer preferences.

Reiterate the value proposition of the MVP

- Our MVP revolutionizes businesses by optimizing sales through personalized recommendations, addressing the Long Tail phenomenon, and fostering a customer-centric shopping experience.

Potential Impact and Benefits of AI Solution



REVENUE GROWTH: THE AI-DRIVEN RECOMMENDATION SYSTEM HAS THE POTENTIAL TO SIGNIFICANTLY BOOST REVENUE BY ENSURING THAT A WIDER ARRAY OF PRODUCTS CONTRIBUTES TO SALES, NOT JUST THE MOST POPULAR ONES.

ENHANCED CUSTOMER EXPERIENCE: AI-RECOMMENDED ITEMS DRIVE GREATER CUSTOMER ENGAGEMENT, LEADING TO INCREASED LOYALTY AND REPEAT BUSINESS.

EFFICIENT MANAGEMENT: TOOLS TO STREAMLINE INVENTORY MANAGEMENT AND WAREHOUSE OPERATIONS, REDUCE BACKLOGS, AND OPTIMIZE RESOURCE ALLOCATION.

References

[1] Recommendation Systems - Stanford InfoLab

[2] Collaborative Filtering - Stanford University

[3] Recommendation systems - Machine Learning - Andrew Ng

[4] Ekstrand, Michael D., John T. Riedl, and Joseph A. Konstan. "Collaborative filtering recommender systems." Foundations and Trends® in Human–Computer Interaction 4.2 (2011): 81-173.

[5] Google Developers, Recommendation Using Deep Neural Networks

[6] Neighborhood-Based Collaborative Filtering

[7] Content-based Recommendation Systems

Thanks!