UC San Diego
JACOBS SCHOOL OF ENGINEERING

# Recommender System

**Team Members: JH (Janghyun) Baek | John Tsai | Justin Shamoun | Muriel Marable | Ying Cui**
**Faculty Advisors: Prof. Julian McAuley, Prof. Ilkay Altintas**

## Introduction

Recommender systems are algorithms that suggest relevant items to users based on data. They generate large revenue for the modern e-commerce industry. 35% of Amazon web sales were generated through their recommended items [source: McKinsey]. This study aims to construct an apparel recommender system for Amazon users through user-rating history, product images and product title text. Multiple deep learning models were built on both readily-available and engineered datasets resulting in a multi-step recommender system. Tableau and a web app are used to display results, along with evaluation measurements.

**Challenge:**

Product recommendations tailored to a user are more likely to lead to higher conversion. Furthermore, users want recommendations of similar items to help discover new products, or compare items. Amazon has millions of clothing products available and for an online shopper finding the right product becomes difficult. The challenge is to create a recommender system for apparel products that is personalized to an Amazon user. Additionally, this recommender system would recommend similar items relative to the item that is currently being viewed. This recommender system will be based on user data, product text and image features. The data streaming/analyzing pipeline has to be scalable to process large datasets in real-time manner for the system to be considered applicable.

**Opportunities as a set of questions:**

Which is the best algorithm to find similarity between users and cluster them and label them? How do we find similarity based on clothing style and how do we measure similarity value? What is the best type of database or tool for information retrieval in order to process the recommender system in real time?

**Approach:**

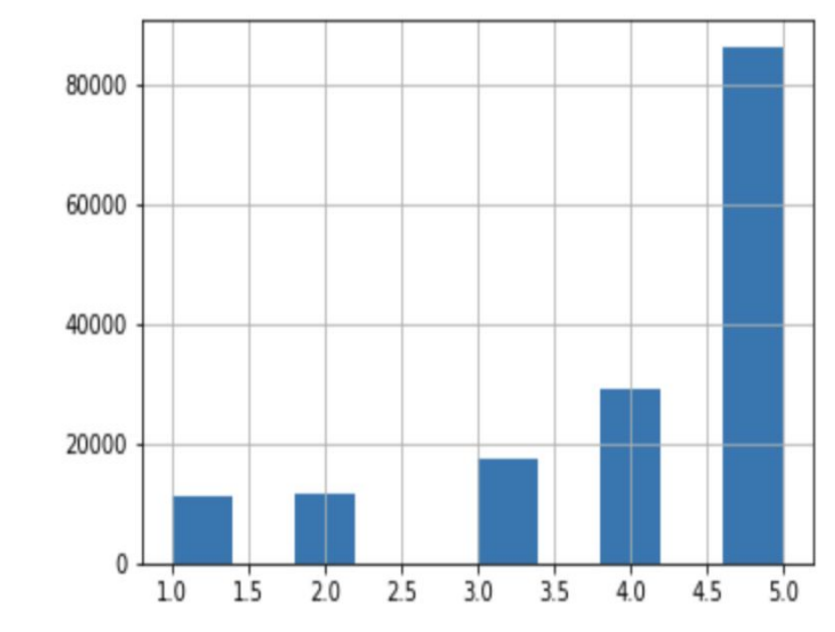
The first approach is content-based recommendation using review text and description text and images of the product. Another approach is active learning where the recommender system will suggest images to the customer and will ask for their input (yes/no). This approach can be used after the implementation of the content-based recommendation. Neo4j graph is going to be used to cross-reference similar products.

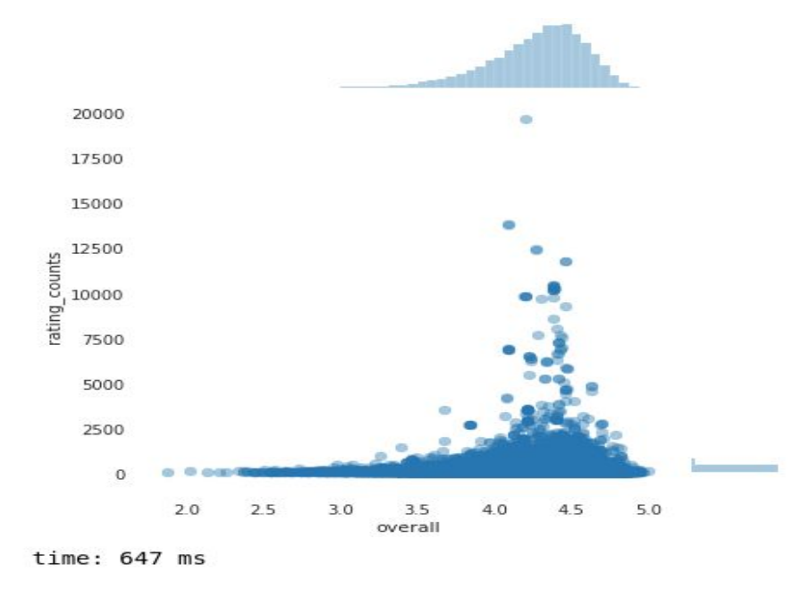
## Data Preparation and EDA (Exploratory Data Analysis)

Amazon Customer Reviews (a.k.a. Product Reviews) is one of Amazon's iconic products. In a period of over two decades since the first review in 1995, millions of Amazon customers have contributed over a hundred million reviews to express opinions and describe their experiences regarding products on the Amazon.com website.

**LINK** - https://s3.amazonaws.com/amazon-reviews-pds/readme.html
**Data Description** - https://s3.amazonaws.com/amazon-reviews-pds/tsv/index.txt

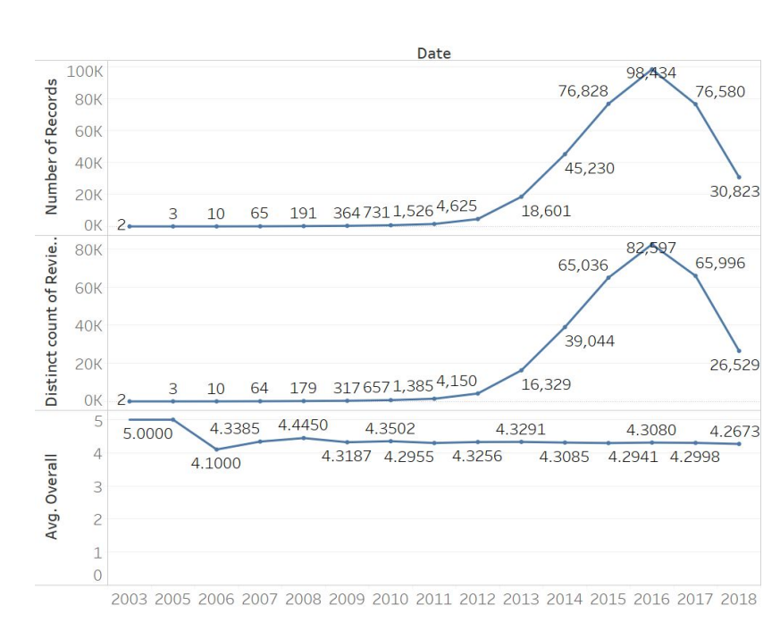
| | |
|---|---|
| **Size** | 61.3 MB |
| **Users** | 94852 |
| **Items** | 97758 |
| **Ratings** | 155509 |



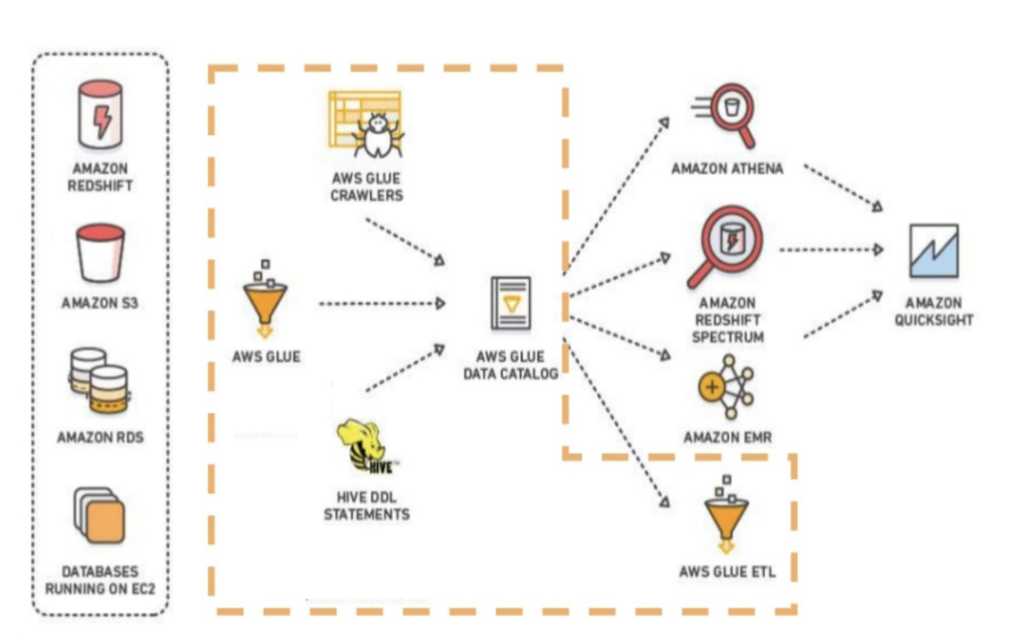


The median in both datasets is 5 rating. This means that the data is skewed towards high ratings.

**Trends over time**



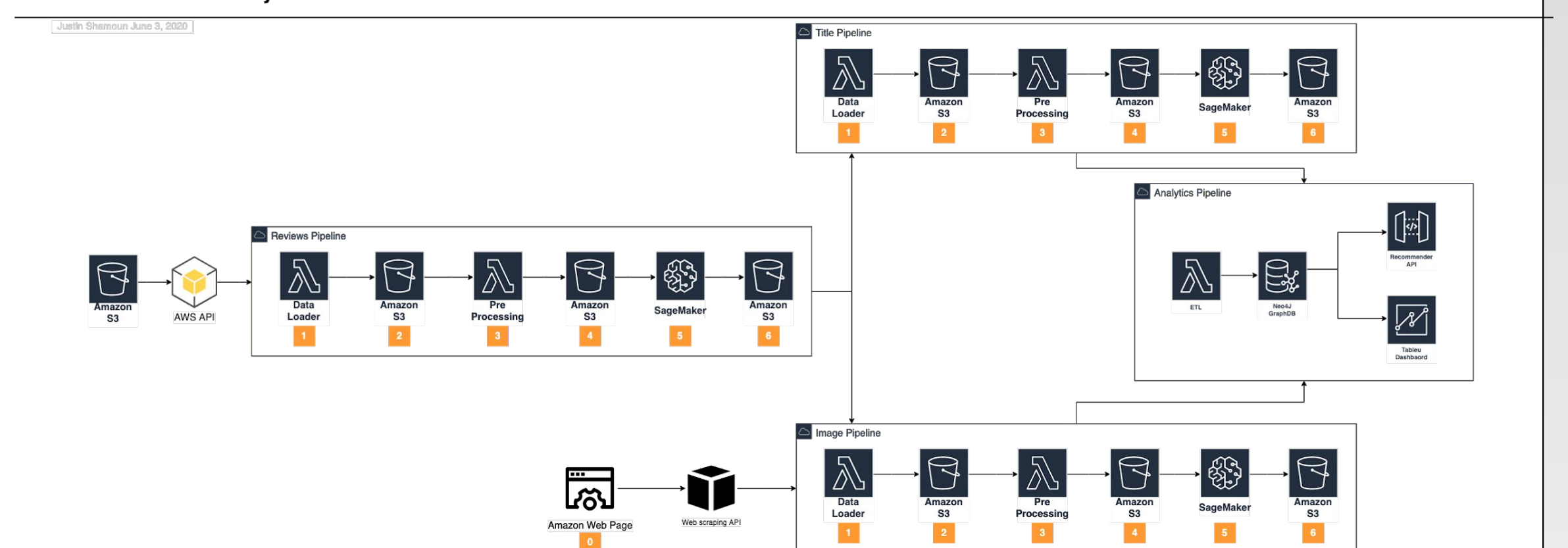
**Using AWS glue**



## Solution Architecture

Our solution uses the following services to deliver recommendations to users in real time and for analytics. Its modular design enables teams to iterate and experiment with a variety of applications and use cases.

**Amazon Recommender System Solution Architecture**

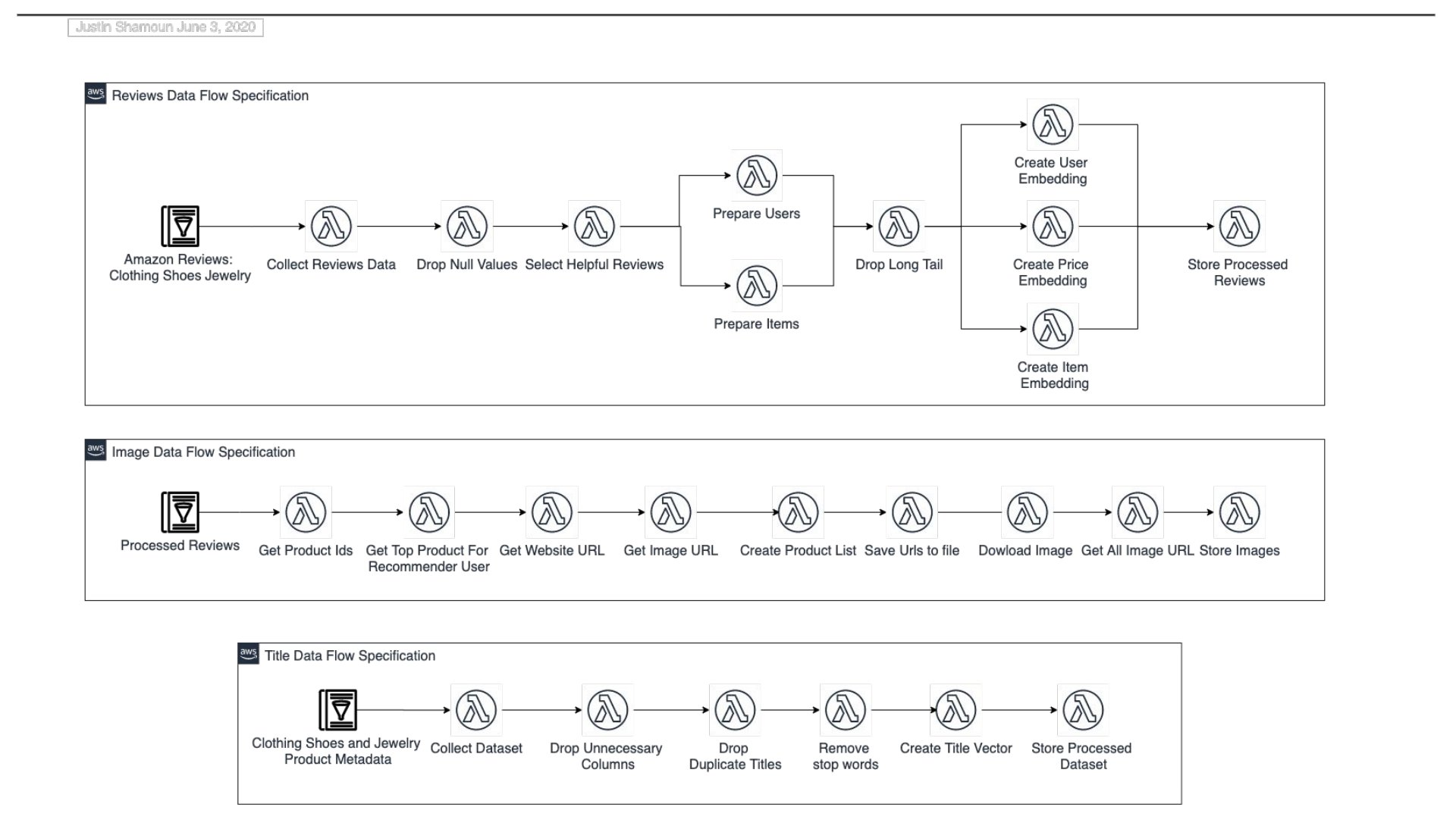


### Data Ingestion, Storage, and Processing

Data is extracted from the Amazon recommender systems data repository and prepared for the initial deep learning pipeline. The data is enriched with images scraped from the Amazon webpage for the second layer of modeling.

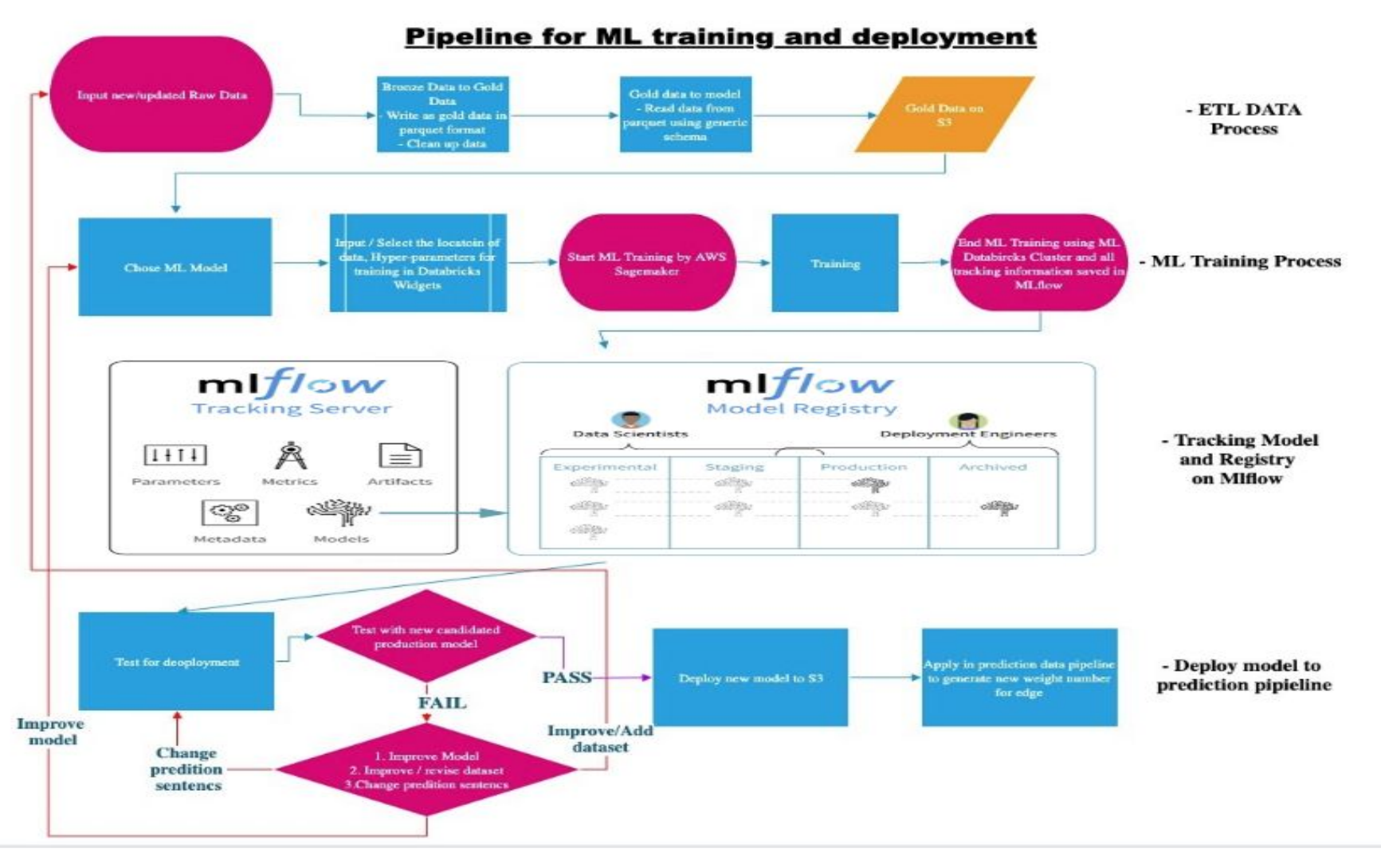
There are three phases of storage as data flows through the system and it is maged with the AWS Glue Catalog. Raw data sets are stored in the landing zone and then moved to standardization once they have been processed. Finally the analytics zone is the location where models store their results. To complete the system, the analytics datasets are stored into a graph database for real time recommendations.

**Amazon Recommender System Data Flow**



Data is transformed and prepared for consumption before each modeling system. Each pipeline includes specialized processing that cater the modeling needs. The pipeline is able to perform these operations at different scales depending on the needs.

### Deployment and Scalability



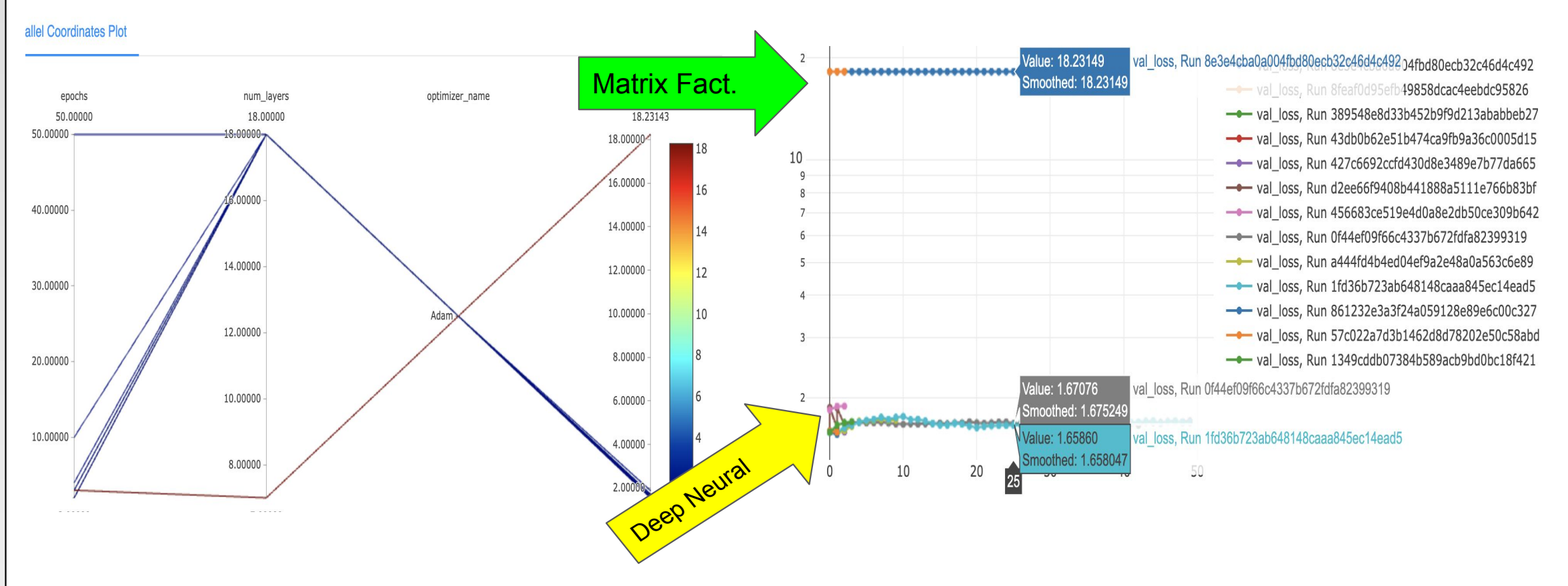
**Scaling relational databases with Apache Spark SQL and DataFrames**

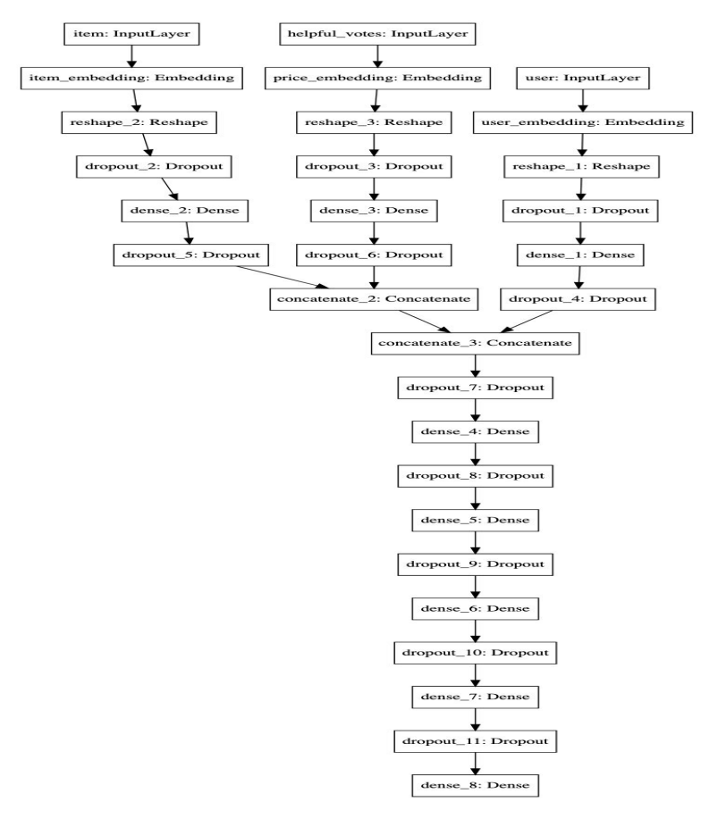
Spark
1 Speed
2 Powerful Caching
3 Deployment
4 Real-Time
5 Polyglot
6 Scalable

## Modeling

### Explicit feedback Recommender Model by Deep Learning

**Compare Matrix Factorization VS Deep neural network**





Created and compared 2 explicit recommendation engines for predicting user's ratings based on 2 machine learning architecture:

- **Matrix Factorization**: Perform a dot product between the respective user and item embeddings.
- **Deep neural network**: Merge user and item embeddings by concatenation or multiplication, and then use them as features for the neural network.

### Image-based Model

We used a pre-trained Deep Learning Convolutional Neural Network model. In particular, we used VGG16 architecture with the Image Net weights with 5 convolutional layers followed by 3 fully-connected layers which has been pre-trained on 1.2 million ImageNet images.We resized images to 224x224x23 pixels and use the output of FC7, the second fully-connected layer, which results in a feature vector of length 4096.

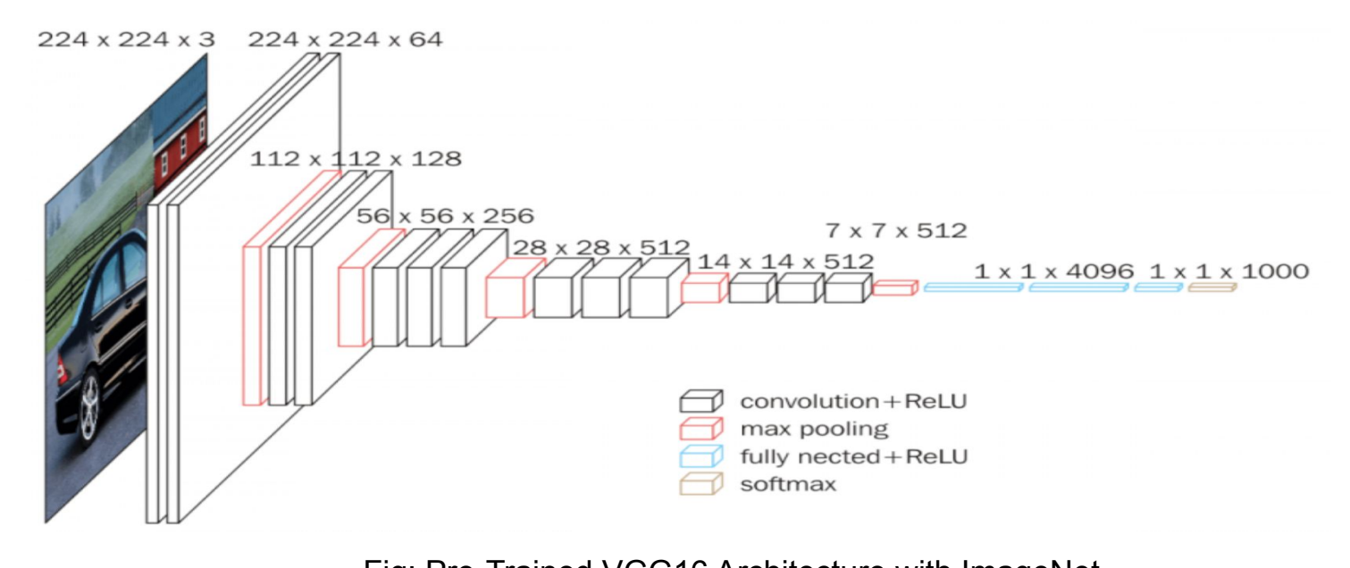


Fig: Pre-Trained VGG16 Architecture with ImageNet

Cosine similarity is used as primary metric to generate the recommendations. For given query product from Deep Learning recommender system results, recommended the top 5 most similar products.

$$Cos\theta = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\ \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2}\ \sqrt{\sum_1^n b_i^2}}$$

where, $\vec{a}\cdot\vec{b} = \sum_1^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ is the dot product of the two vectors.

### Natural Language Processing Model

We used a TF-IDF weighted word2vec model on the product title text and brand text. Product title text was pre-processed by removing stop words and removing duplicates. TF-IDF is computed for the corpus. Word2vec is used to produce word embeddings, with the TF-IDF values acting as weights of the words, which determines the strength of each input connected to a given neuron.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

Equation: TF-IDF

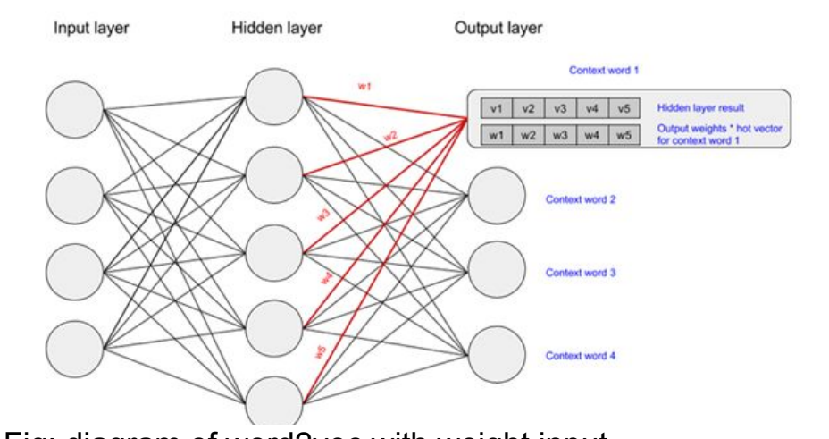
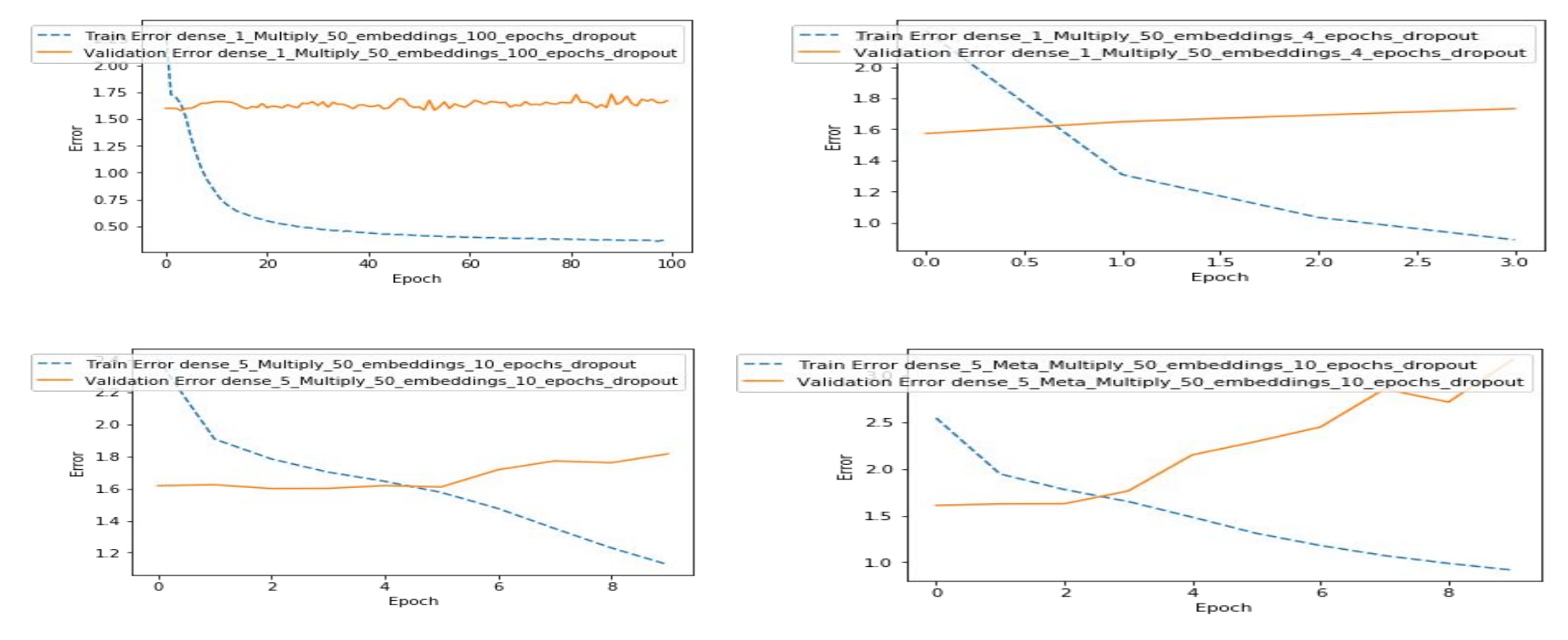


Fig: diagram of word2vec with weight input

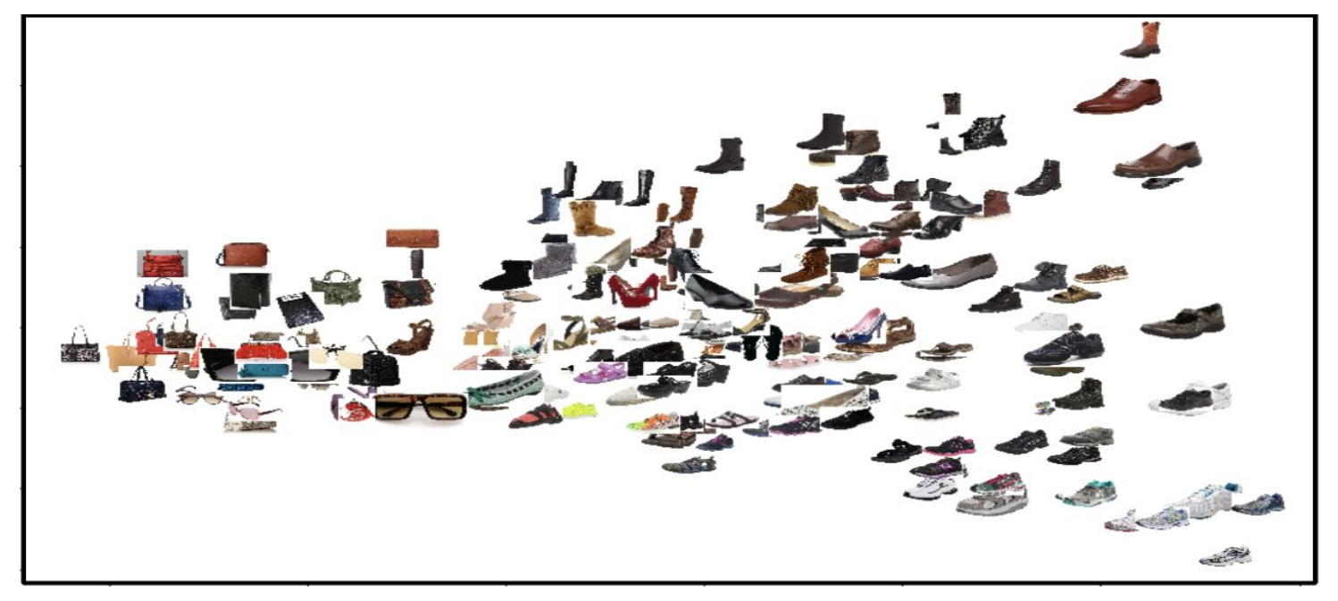
Word2vec is used to produce word embeddings, with the TF-IDF values acting as weights of the words, which determines the strength of each input connected to a given neuron.

Euclidean distance is used to compute the similarity between the generated word vectors. The top 5 with the closest distance to the given product are chosen as the recommended items.

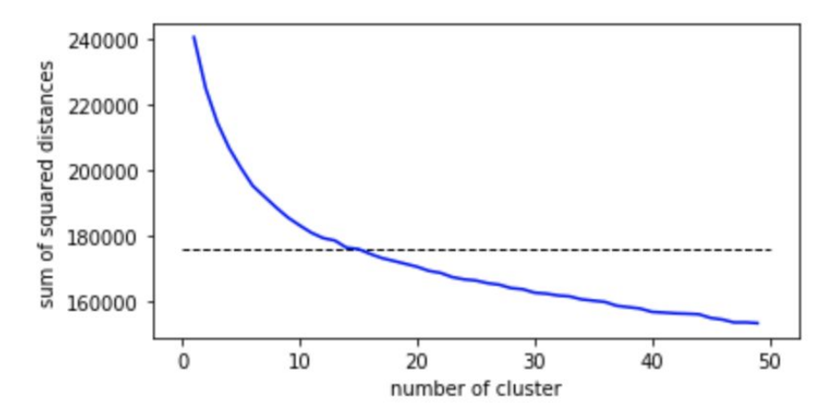
## Key Finding and Results

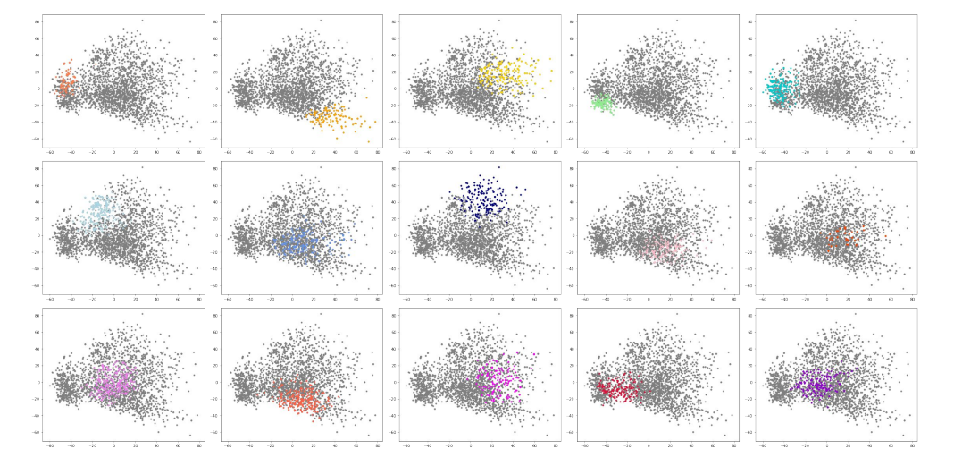


- I showed that using deep neural networks can achieve better performance than using matrix factorization.
- Going deeper (more than 3 layers) seems to lead to overfitting and not to further improvement.
- Adding epochs, reducing embedding size or change hidden units numbers does not help either.
- Running on a larger dataset does not help either, because the data in both datasets is very skewed.



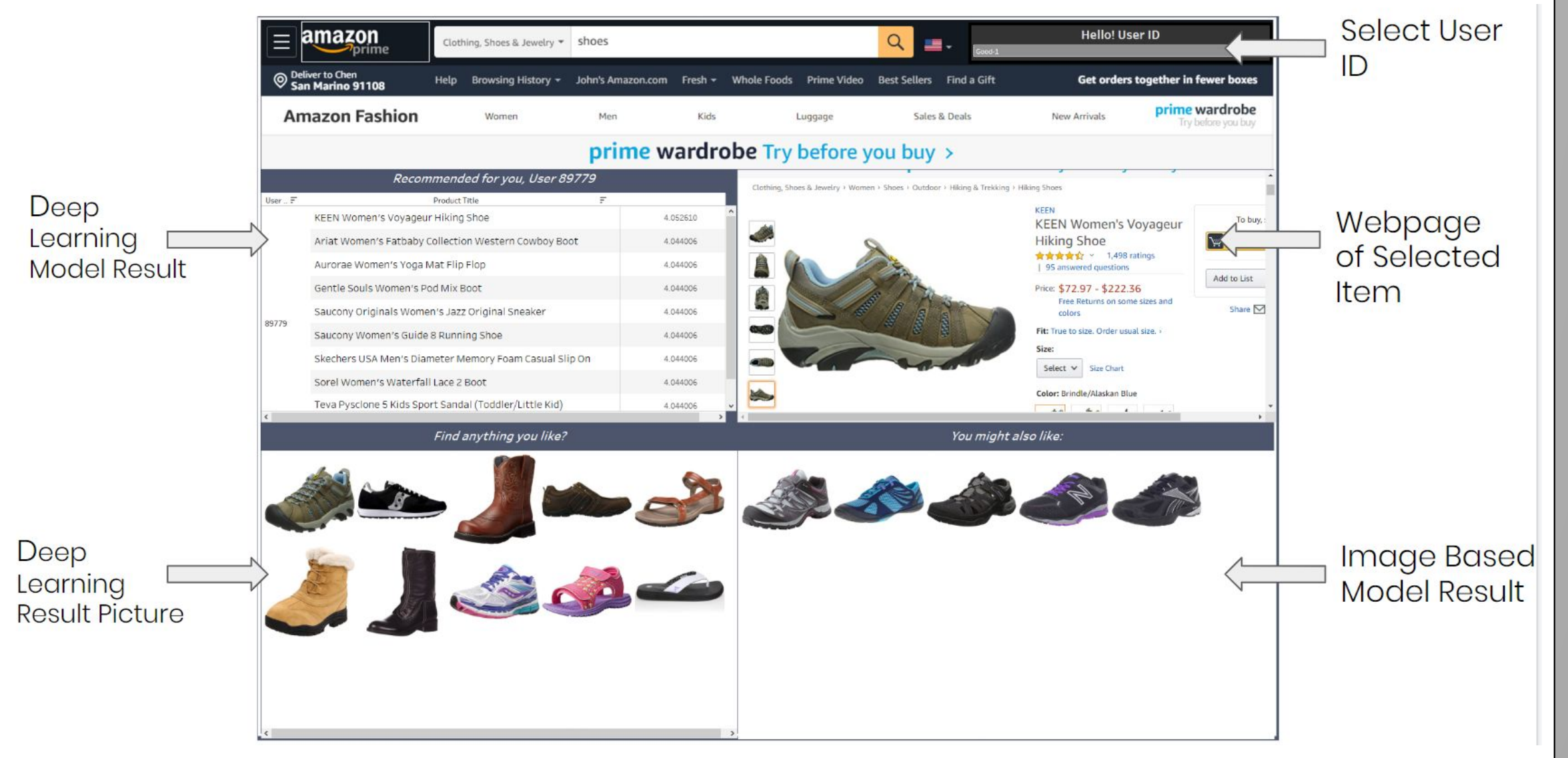
- Used the features extracted from CNN model visualized two-dimensional embeddings of sample of shoes. Sunglasses drifts gradually toward bags and slippers and sandals drift smoothly towards sporting shoes.





- Used K-Means algorithm and elbow method to determine proper number of clusters. Analyzed the sum of the inertia of model up to 50 clusters.
- Visualized two-dimensional projection highlighting the products that belong to each of the 15 clusters.

## Dashboard



The Tableau interactive dashboard was designed to visualize various model output with actual picture. This creates a view that assimilate how the output would look like from a customer's perspective when actual deployment happens. The two panels on the left illustrate the top 10 recommended product per given user based on neural network model. Bottom right panel shows the recommended from image-based model. Top right panel is a dynamic webpage that navigate user to the current website of the selected product.

## Conclusion

- Data engineering on selective data was more effective than tuning parameters on various models.
- The usage of cloud-computing tools such as Amazon SageMaker and Databricks enables quick workflow to provide effective model building and testing.
- Deep learning model with appropriate drop-outs implemented had the best performance in terms of accuracy.
- Tableau dashboard provides a quick view on how model outputs would look like from a customer's view.