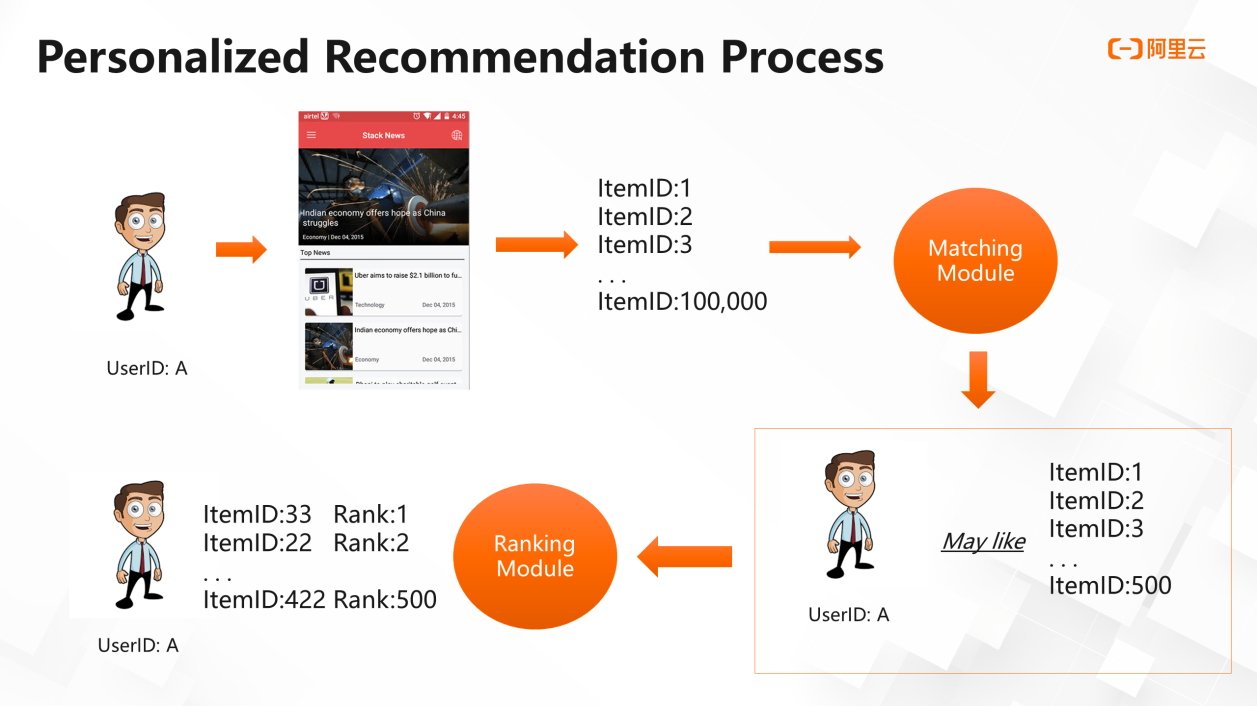
Personalized recommender system

**Recommendation engines are a subclass of machine learning that generally rank or rate products/users**.

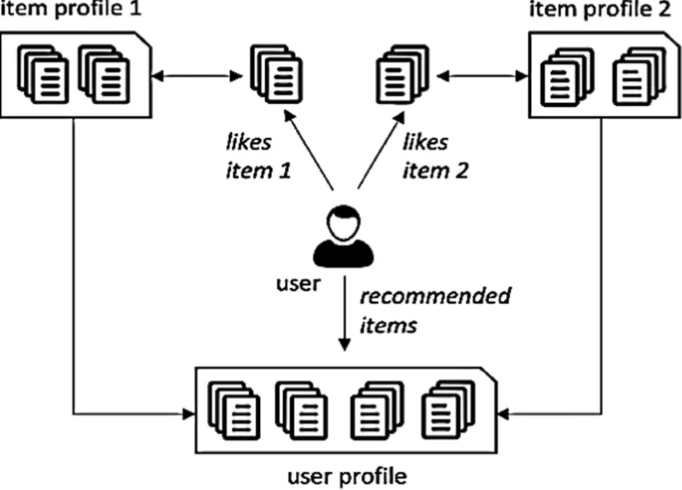
Good recommendations for example Video on Demand (VOD) platforms can increase revenue for long tail content by surfacing it in recommendations based on consumers’ behavior

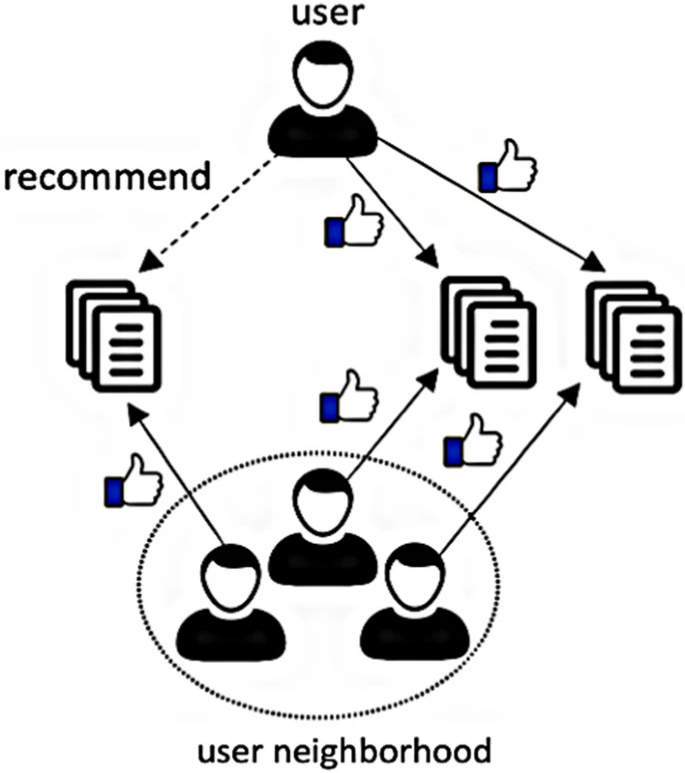


Filtering the content:-

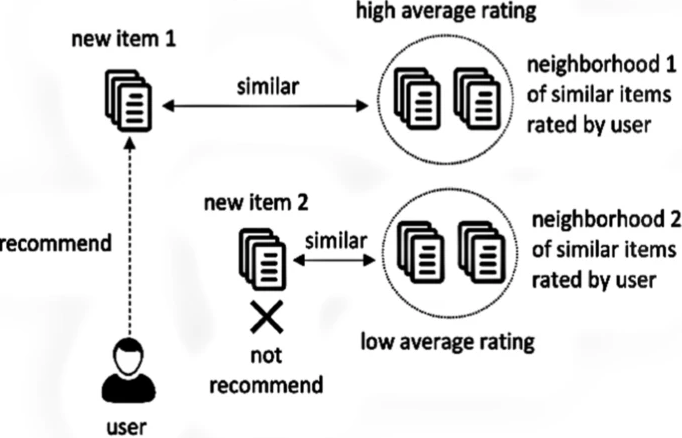
Content-based filtering is one of the simplest systems, but sometimes is still useful. It is based on known user preferences provided explicitly or implicitly

Collaborative filtering is based on (user, item, rating) tuples. So, unlike content-based filtering, it leverages other users’ experiences.



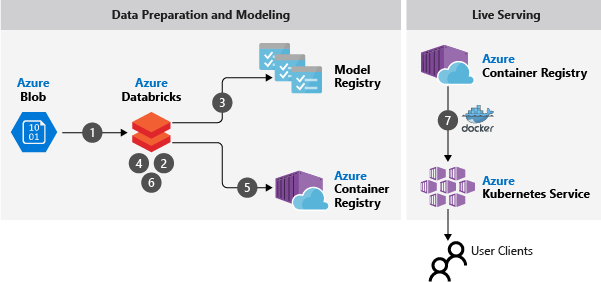


**Fig 2**



**Fig 3**

**Machine learning sample model:**



**Dataflow:**

When the data is available, the following steps are taken to build and operationalize a recommendation system:

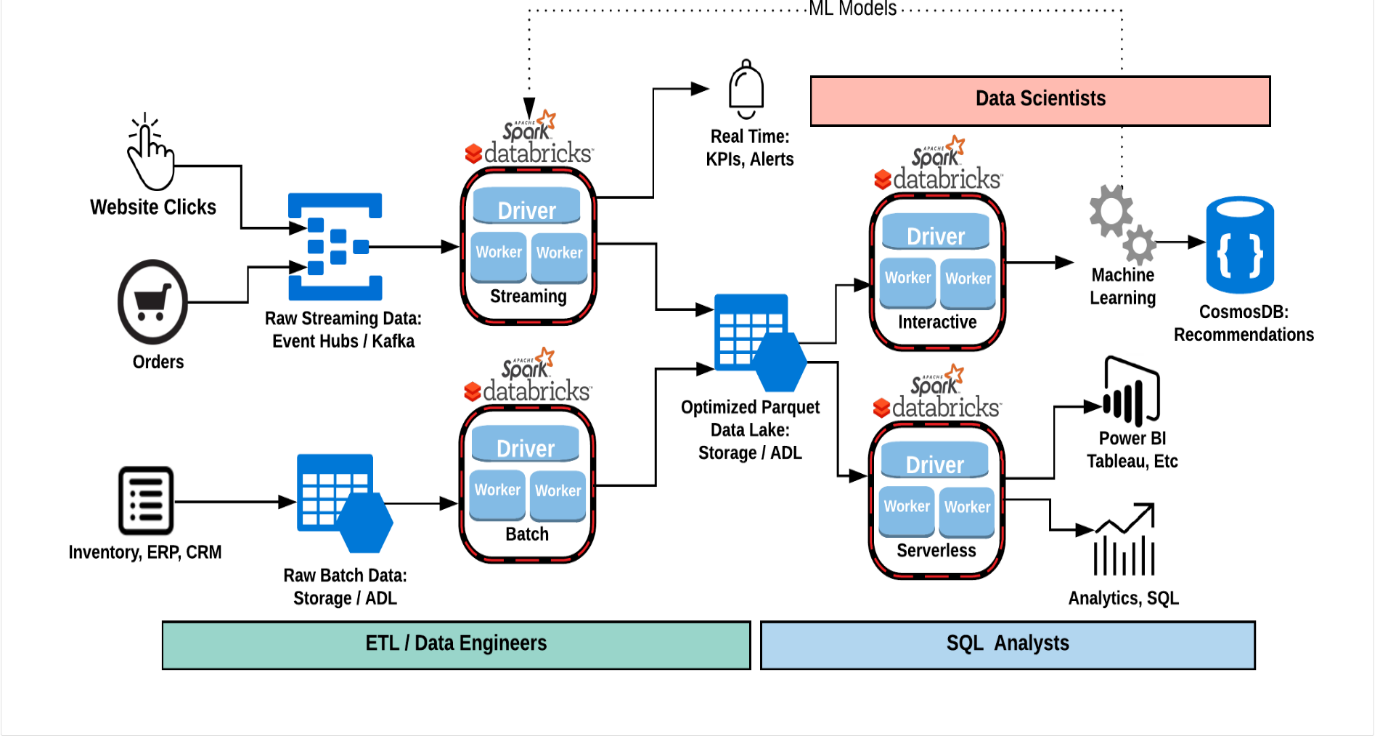
1. The sets of distinct user and item data are pre-processed and joined, which results in a mixture of numeric and categorical features to be used for predicting user-item interactions (clicks). This table is uploaded to [Azure Blob Storage](https://docs.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction).
2. The [MMLSpark](https://aka.ms/spark) library enables the training of a [LightGBM](https://github.com/Microsoft/LightGBM) classifier on Azure Databricks to predict the click probability as a function of the numeric and categorical features that were created in the previous step. LightBGM is a highly efficient machine learning algorithm, and MMLSpark enables the distributed training of LightGBM models over large datasets.
3. The trained classifier is serialized and stored in the Azure Model Registry. With Azure Model Registry, you can store and organize different versions of the model (for example, based on newer data or different hyperparameters) within an Azure Machine Learning workspace.
4. A serving script is defined by using the [MML Spark Serving](https://mmlspark.blob.core.windows.net/website/index.html) library to provide predictions from the trained model.
5. Machine Learning is used to create a Docker image in [Azure Container Registry](https://docs.microsoft.com/en-us/azure/container-registry) that holds the image with the scoring script and all necessary dependencies for serving predictions.
6. Machine Learning is also used to provision the compute for serving predictions. A Kubernetes cluster is configured by using [Azure Kubernetes Service (AKS)](https://docs.microsoft.com/en-us/azure/aks/intro-kubernetes) with the number of nodes that are needed to handle the expected load. The virtual machine (VM) size can be adjusted based on the model's computation and memory requirements.
7. The scoring service is deployed as a web service on the AKS cluster. The service provides an endpoint where user and item features can be sent to receive the predicted probability of a click for that user and item.

**Components**

This architecture makes use of the following components:

* [Blob Storage](https://azure.microsoft.com/services/storage/blobs) is a storage service that's optimized for storing massive amounts of unstructured data. In this example scenario, the input data is stored here.
* [Azure Databricks](https://azure.microsoft.com/services/databricks) is a managed Apache Spark cluster for model training and evaluation. The scenario also uses [MMLSpark](https://aka.ms/spark), a Spark-based framework that's designed for large-scale machine learning.
* [Container Registry](https://azure.microsoft.com/services/container-registry) is used to package the scoring script as a container image, which is used to serve the model in production.
* [AKS](https://azure.microsoft.com/services/kubernetes-service) is used to deploy the trained model to web or app services.
* [Machine Learning](https://azure.microsoft.com/services/machine-learning-service) is used in this scenario to register the machine learning model and to deploy AKS.
* [Microsoft Recommenders](https://github.com/Microsoft/Recommenders) is an open-source repository that contains utility code and samples. By using this repository, users can start to build, evaluate, and operationalize a recommender system.

**Complete generalised process:**



### The Data Interfaces

There are several key interfaces that you should understand when you go to use Spark.

**Dataset**

* The Dataset is Apache Spark's newest distributed collection and can be considered a combination of DataFrames and RDDs. It provides the typed interface that is available in RDDs while providing a lot of conveniences of DataFrames. It will be the core abstraction going forward.

**DataFrame**

* The DataFrame is collection of distributed Row types. These provide a flexible interface and are similar in concept to the DataFrames you may be familiar with in python (pandas) as well as in the R language.

**RDD (Resilient Distributed Dataset)**

* Apache Spark's first abstraction was the RDD or Resilient Distributed Dataset. Essentially it is an interface to a sequence of data objects that consist of one or more types that are located across a variety of machines in a cluster. RDD's can be created in a variety of ways and are the "lowest level" API available to the user. While this is the original data structure made available, new users should focus on Datasets as those will be supersets of the current RDD functionality.

## Databricks Runtime & Spark Clusters

### Azure Databricks is designed for Azure! This means:

* Decoupling Storage and Compute
* Ephemerial Clusters
* Multiple Clusters
* Autoscaling / Serverless

### Azure Databricks Clusters:

* Clusters Spin up in minutes (~5min) :-)
* Two types of clusters: Interactive (shared) /Job Clusters
* Interactive clusters can also be Azure Databricks Serverless Pools
* SQL Endpoints (JDBC/ODBC) for Power BI, Tableau, etc.

## Databricks Workspace & Notebooks

* Interactive notebooks with support for multiple languages (SQL, Python, R and Scala)
* Real-time collaboration
* Notebook revision history and version control
* One-click visualizations
* Workspace ACLs
* Library Management
* Ability to convert a notebook into a Job