

# FROM A PYTHON APPLICATION TO A MACHINE LEARNING PREDICTION WITH SPARK

MAY 2020

TECH THURSDAY



1

About the Demo

2

Recommendation  
Pipeline

3

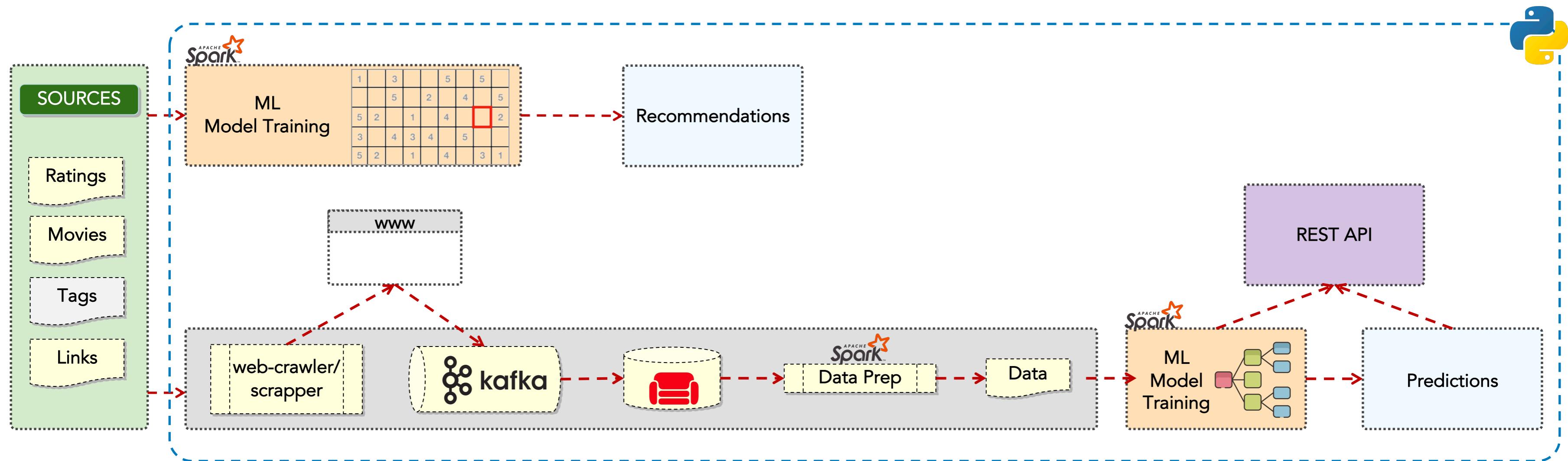
Prediction  
Pipeline

4

REST API on Top  
of Spark ML  
Models

# WHAT WE WILL SEE

Why this talk?



# ABOUT THE DATA

From where these data came from

## top picks

see more

based on your ratings, MovieLens recommends these movies

Band of Brothers 2001   R   705 min	Casablanca 1942   PG   102 min	One Flew Over the Cuckoo's Nest 1975   R   133 min	The Lives of Others 2006   R   137 min	Sunset Boulevard 1950   NR   110 min	The Third Man 1949   NR   104 min	Path to Glory 1957   R   100 min
--	-----------------------------------	---	---	---	--------------------------------------	-------------------------------------

Recent releases

see more

movies released in last 90 days that you haven't rated

Cantinflas 2014   PG   106 min	Felony 2014   R   106 min	What If 2014   PG-13   102 min	Frank 2014   R   96 min	Sin City: A Dame to Kill For 2014   R   102 min	If I Stay 2014   PG-13   106 min	Are We There Yet? 2014   PG   96 min
-----------------------------------	------------------------------	-----------------------------------	----------------------------	--	-------------------------------------	---

IMDb

All | Search IMDb

TRYING

Watch on the Apple TV app.

Try it free

FULL CAST AND CREW | TRIVIA | USER REVIEWS | IMDbPro | MORE

SHARE

Toy Story (1995)

PG | 1h 21min | Animation, Adventure, Comedy | 22 March 1996

8.3/10 850,105 Rate This

Watch on Prime Video rent/buy from GBP2.49

+ Add to Watchlist

Metascore 95 From metacritic.com

Reviews 585 user | 159 critic

Popularity 613 (↑ 27)

Everything New on Netflix in May

No need to waste time endlessly browsing—here's the entire lineup of new movies and TV shows streaming on Netflix this month.

See the full list

Related News

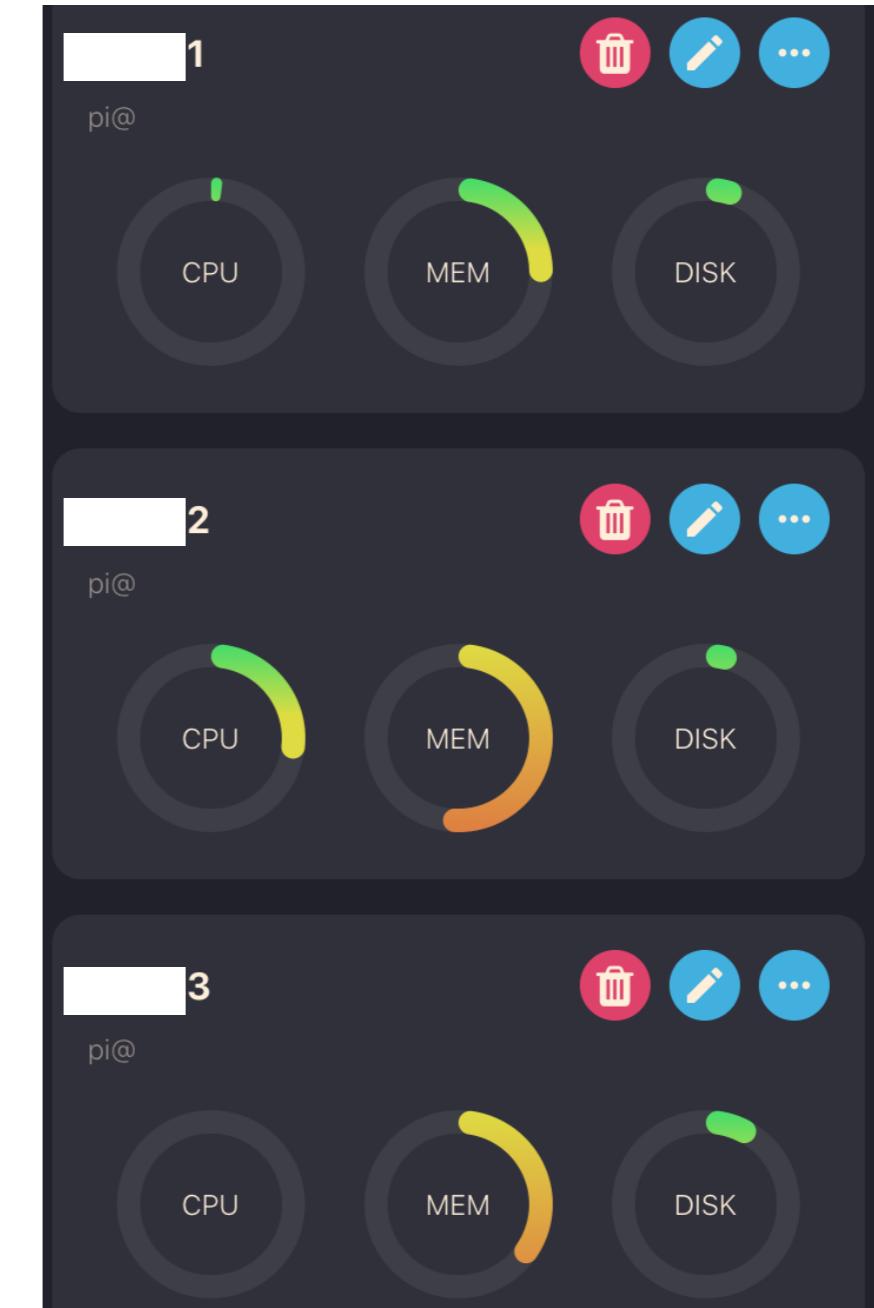
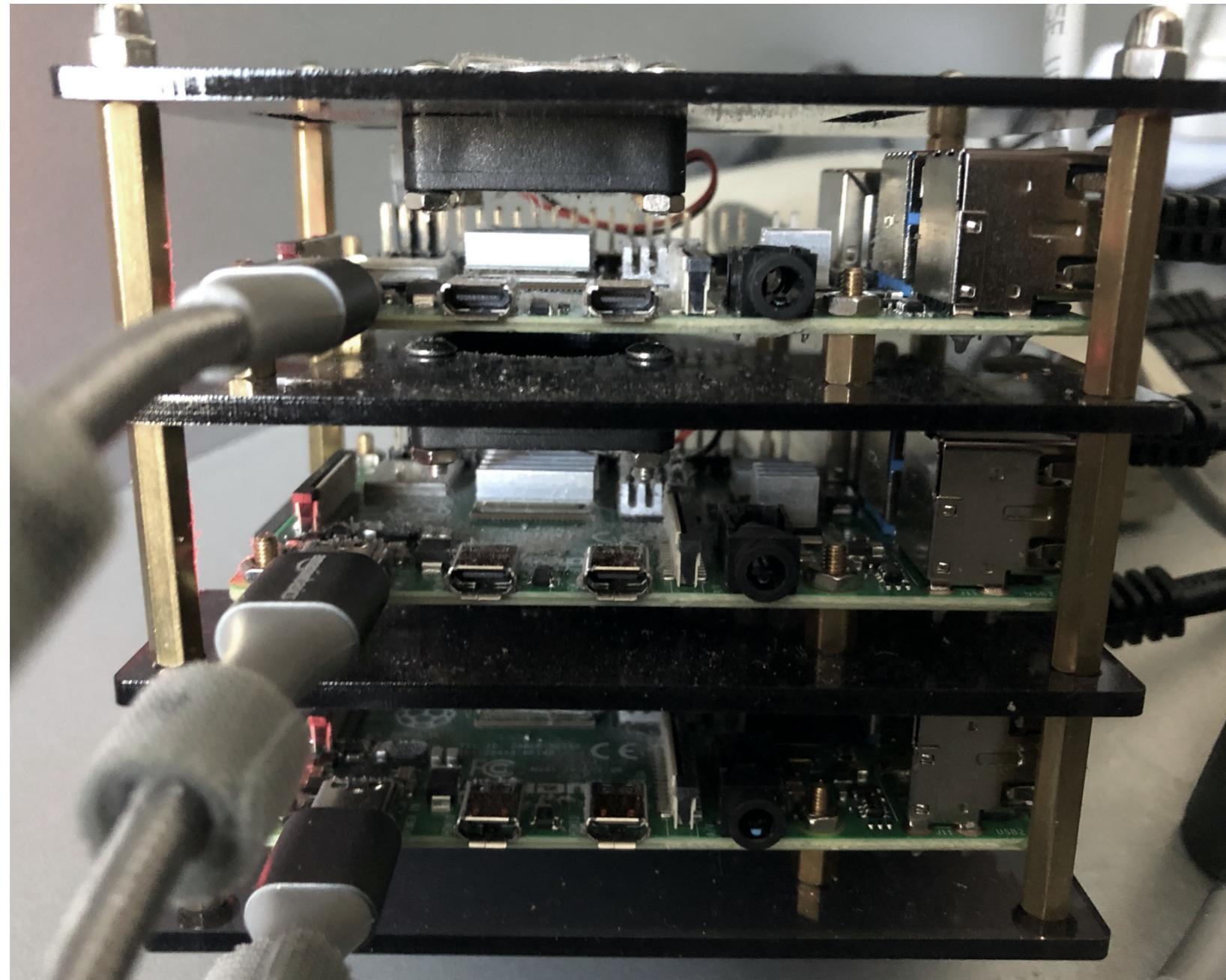
Seth Rogen, Evan Goldberg Developing Adult Animated Comedy 'Bubble' Based on Podcast at Sony

Disney Brands Cloth Face Masks with Iconic Characters, Including Baby Yoda

'Transformers': 'Toy Story 4' Director Josh Cooley To Helm An Animated

# THE USED CLUSTER

## Raspberry Pi Cluster



# SPARK ML

## Version 2.4.4

The screenshot shows the Apache Spark 2.4.4 MLlib API documentation. At the top, there's a navigation bar with links for Overview, Programming Guides, API Docs, Deploying, and More. Below the navigation is a sidebar with two main sections: "MLlib: Main Guide" and "MLlib: RDD-based API Guide". The "MLlib: Main Guide" sidebar lists topics like Basic statistics, Data sources, Pipelines, Extracting, transforming and selecting features, Classification and Regression, Clustering, Collaborative filtering, Frequent Pattern Mining, Model selection and tuning, and Advanced topics. The "MLlib: RDD-based API Guide" sidebar lists topics such as Data types, Basic statistics, Classification and regression, Collaborative filtering, Clustering, Dimensionality reduction, Feature extraction and transformation, Frequent pattern mining, Evaluation metrics, PMML model export, and Optimization (developer). The main content area contains code examples for ChiSquareTest. It includes a note to refer to the Scala docs for details, a code block with Scala code for performing a Chi-Square test on a DataFrame, and a link to the full example code in the Spark repository. Below this, there's a section for Summarizer with tabs for Scala, Java, and Python, followed by a code example for using Summarizer on a DataFrame.

Refer to the [ChiSquareTest Scala docs](#) for details on the API.

```
import org.apache.spark.ml.linalg.{Vector, Vectors}
import org.apache.spark.ml.stat.ChiSquareTest

val data = Seq(
  (0.0, Vectors.dense(0.5, 10.0)),
  (0.0, Vectors.dense(1.5, 20.0)),
  (1.0, Vectors.dense(1.5, 30.0)),
  (0.0, Vectors.dense(3.5, 30.0)),
  (0.0, Vectors.dense(3.5, 40.0)),
  (1.0, Vectors.dense(3.5, 40.0))
)

val df = data.toDF("label", "features")
val chi = ChiSquareTest.test(df, "features", "label").head
println(s"pValues = ${chi.getAs[Vector](0)}")
println(s"degreesOfFreedom ${chi.getSeq[Int](1).mkString("[", ", ", ", ])}")
println(s"statistics ${chi.getAs[Vector](2)}")
```

Find full example code at "examples/src/main/scala/org/apache/spark/examples/ml/ChiSquareTestExample.scala" in the Spark repo.

## Summarizer

We provide vector column summary statistics for Dataframe through [Summarizer](#). Available metrics are the column-wise max, min, mean, variance, and number of nonzeros, as well as the total count.

[Scala](#) [Java](#) [Python](#)

The following example demonstrates using [Summarizer](#) to compute the mean and variance for a vector column of the input dataframe, with and without a weight column.

```
import org.apache.spark.ml.linalg.{Vector, Vectors}
import org.apache.spark.ml.stat.Summarizer

val data = Seq(
  (Vectors.dense(2.0, 3.0, 5.0), 1.0),
  (Vectors.dense(4.0, 6.0, 7.0), 2.0)
)

val df = data.toDF("features", "weight")

val (meanVal, varianceVal) = df.select(metrics("mean", "variance")
  .summary($"features", $"weight").as("summary"))
  .select("summary_mean", "summary_variance")
```

# **Recommendation Pipeline**

# A LITTLE BIT ABOUT RECOMMENDATION SYSTEMS

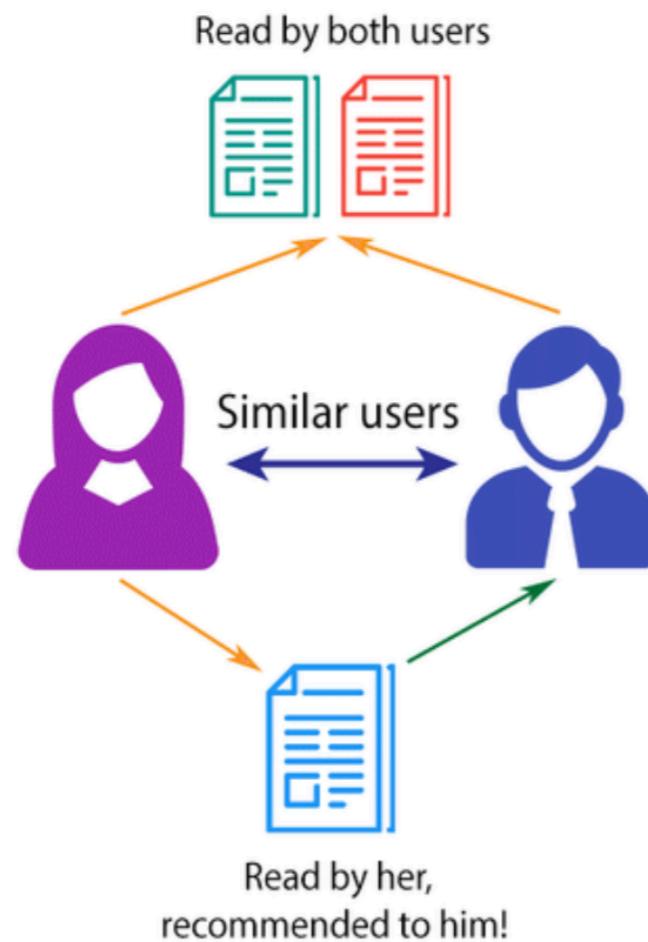
Recommendation References:



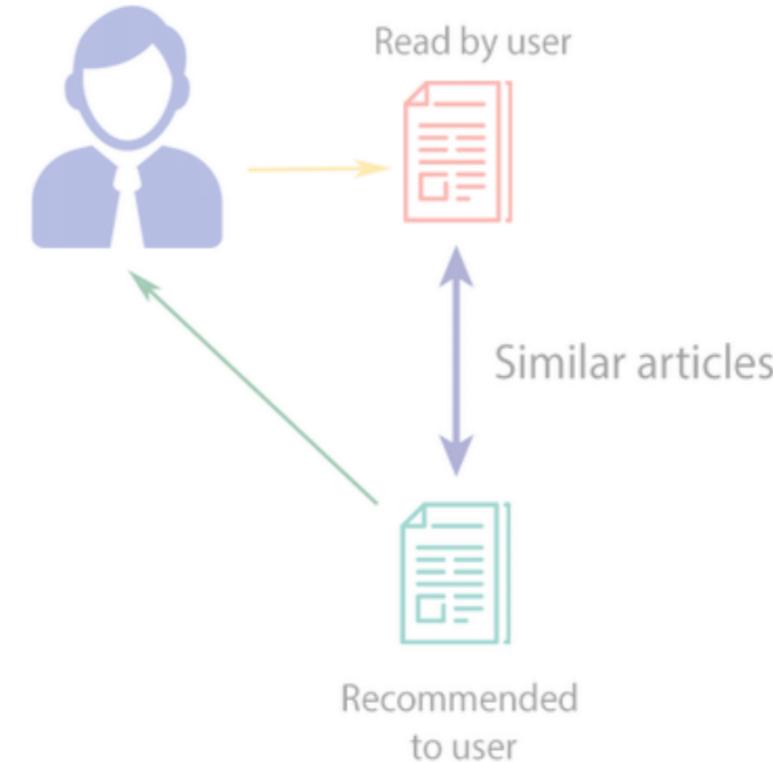
amazon

NETFLIX

## COLLABORATIVE FILTERING



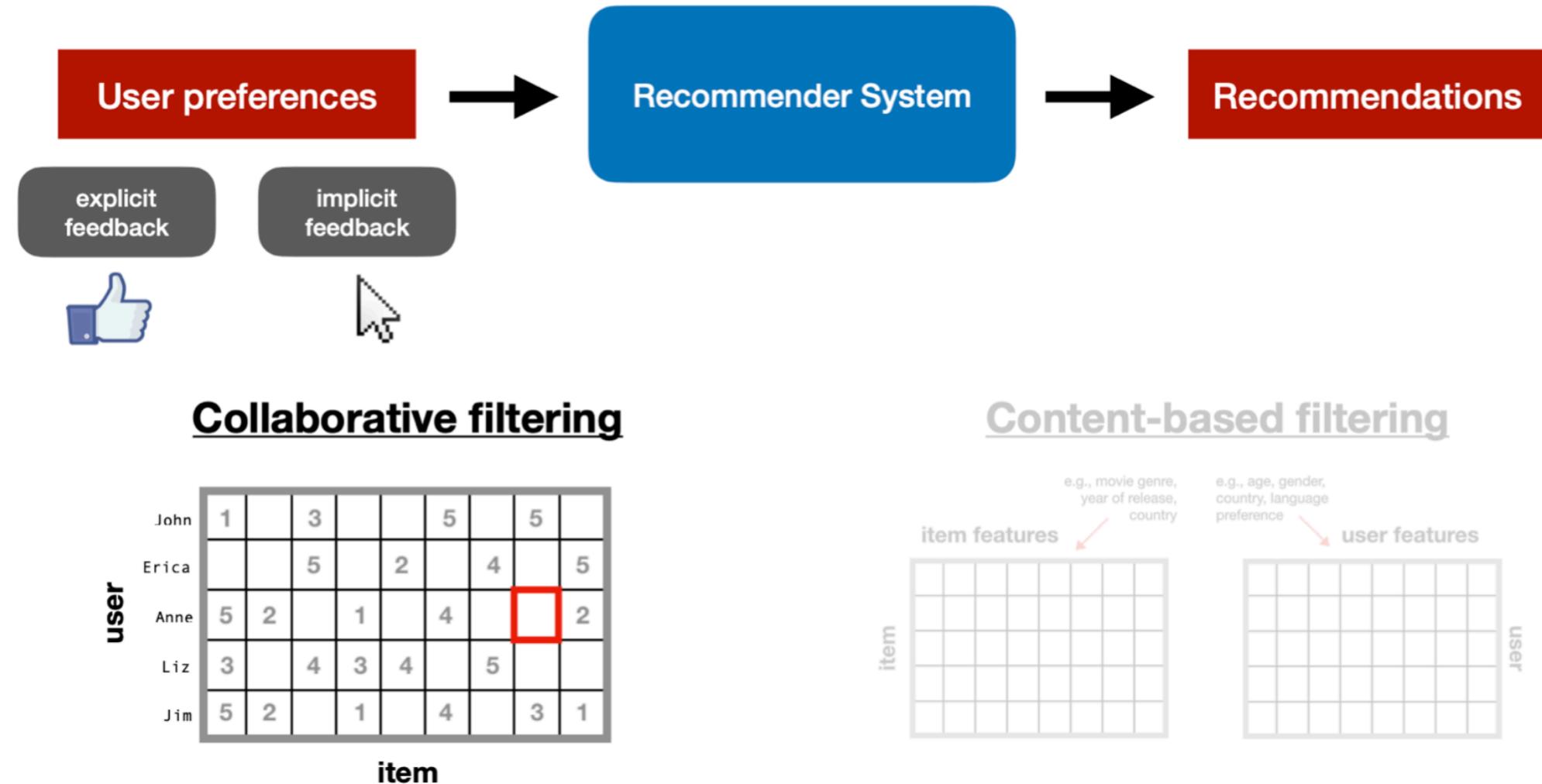
## CONTENT-BASED FILTERING



Spark ML: Alternating Least Square (ALS)

# A LITTLE BIT ABOUT RECOMMENDATION SYSTEMS

How it works?



# A LITTLE BIT ABOUT RECOMMENDATION SYSTEMS

## Collaborative Filtering: Factorization Matrix

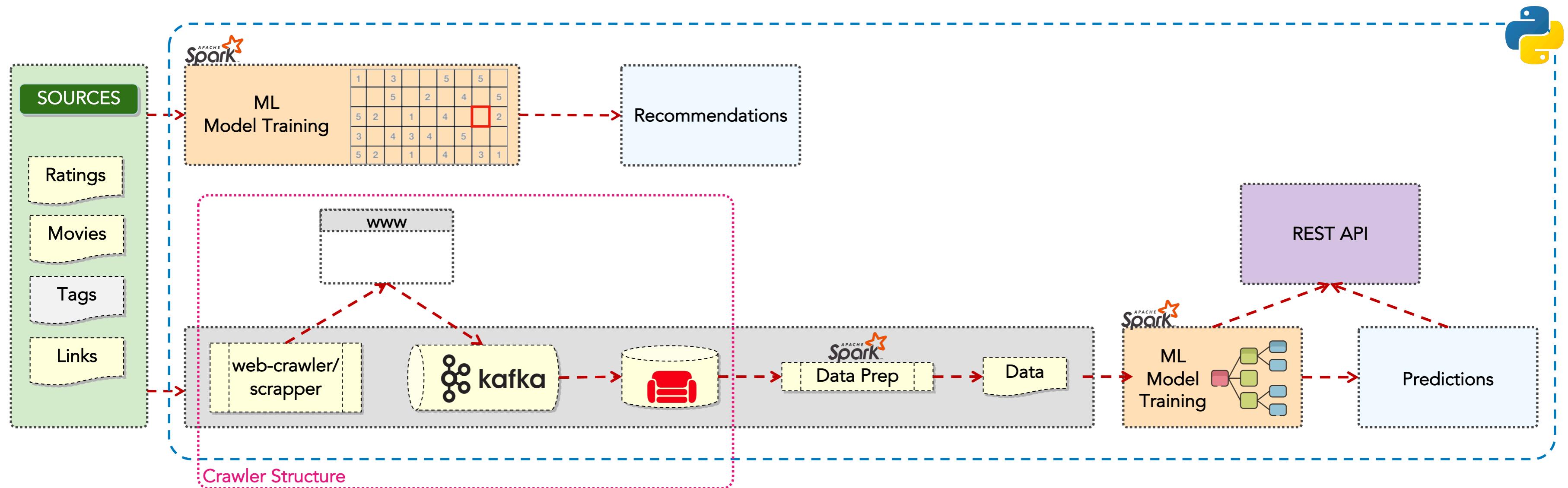
	Item1	Item2	Item3	Item4
User1	x	x		x
User2		x	x	
User3				x
User4	x	<b>x</b>	x	

Let's see an example with some theory

# **Prediction Pipeline**

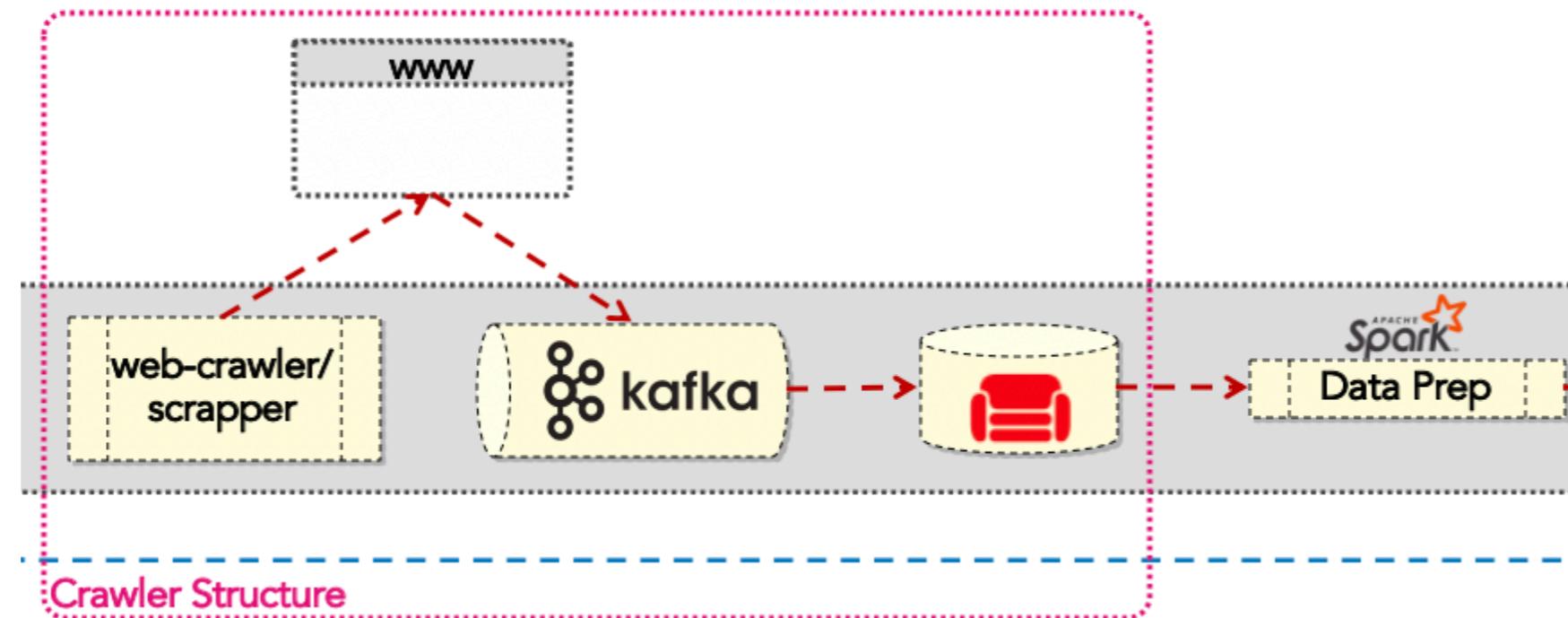
# LET'S LOOK IT AGAIN

## The prediction pipeline



# THE CRAWLER STRUCTURE AND THE DATA PREPARATION

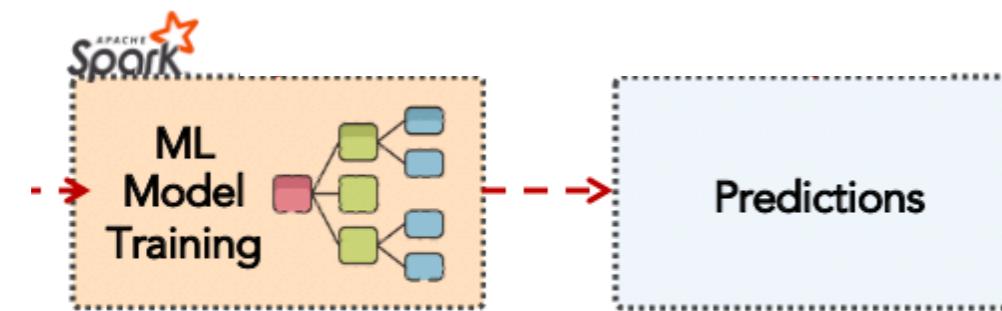
Understanding the process: The objective is to get more data about the movies.



Let's take a look on it!

# PREDICTION MODEL

Predict the budget value for each movie based on the Director, Genre and Rating



The screenshot shows the Apache Spark 2.4.4 documentation page. The header includes the Apache Spark logo, version 2.4.4, and navigation links for Overview, Programming Guides, API Docs, and Deploying.

The main content area features two sections:

- MLlib: Main Guide**: A list of various machine learning components including Basic statistics, Data sources, Pipelines, Feature extraction, Classification and Regression, Clustering, Collaborative filtering, Frequent Pattern Mining, Model selection and tuning, and Advanced topics.
- MLlib: RDD-based API Guide**: A list of RDD-based API components including Data types, Basic statistics, and Classification and Regression.

A red dashed box highlights the "Available families" section under the Regression heading, which lists Decision tree regression, Random forest regression, Gradient-boosted tree regression, Survival regression, and Isotonic regression.

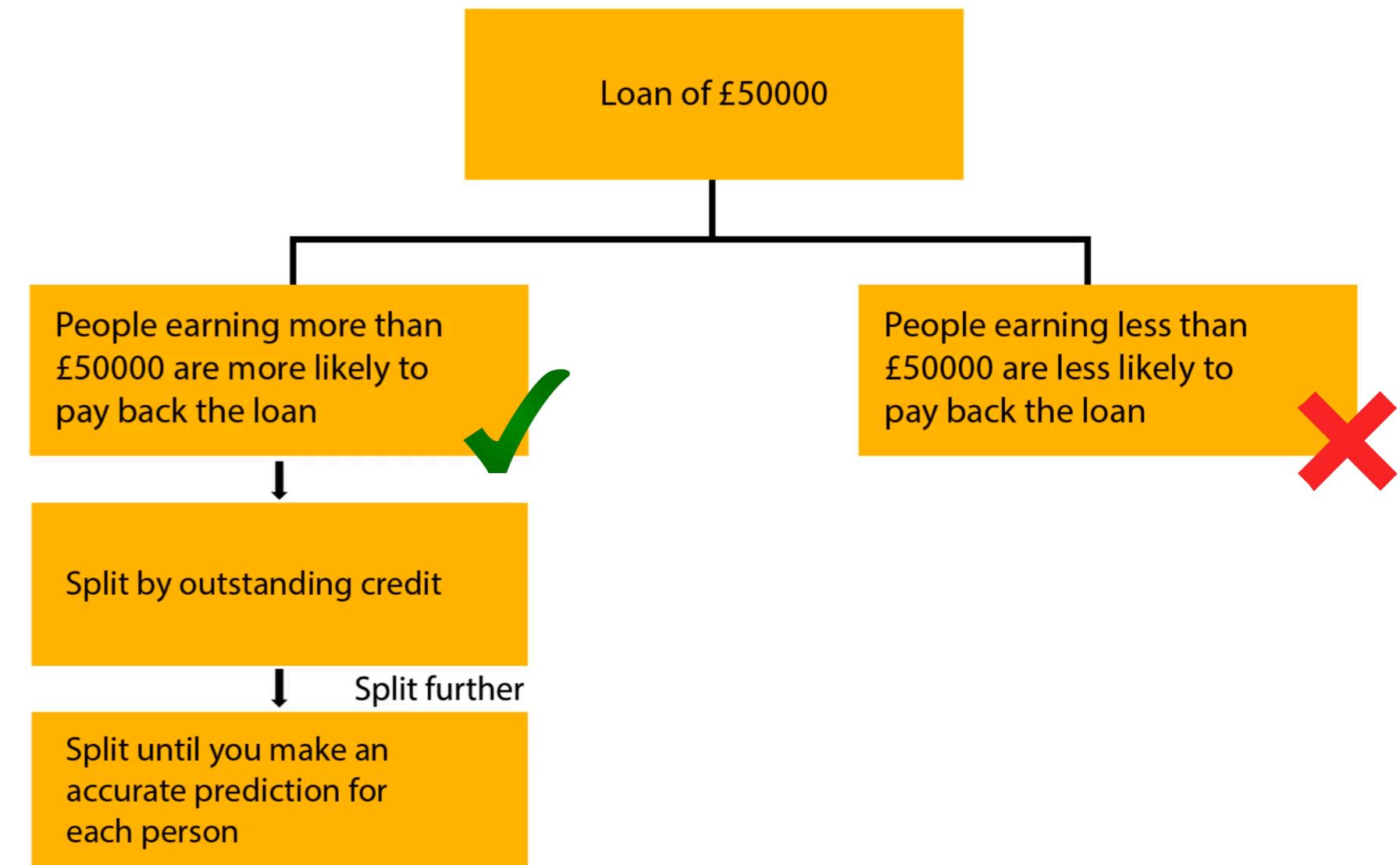
# PREDICTION MODEL

A little bit about the model theory

## Decision Tree

- It starts with all the data and then it tries to find the best split up the individuals such that we've got a better prediction for both groups.
- It can be used to predict a number or to predict the probability of an event.
- Decision Tree models are very sensible if there is/are update(s) in the Analytical Base Table (ABT).
- Decision Tree Example: <http://www.r2d3.us/visual-intro-to-machine-learning-part-1/>

## Example: bank loan repayment



# PREDICTION MODEL

## A little bit about the model theory

### **Random Forest**

- Split data into thousands of random subsamples.
- Build a simple decision tree, for example tree leaves deep.
- Obtain a thousand each with a different prediction.
- Average all the decisions.
- The average prediction of many simple trees is more accurate than the single prediction of a very complicated tree.

### **Gradient Boosted**

- It follows same Random Forest concept, but it applies some weights “scoring”.
- Combine trees at the beginning, instead of the end.
- Build one tree at time.

# SPARK FEATURE EXTRACTION

## Continuous Features Transformers

- Standard Scaler
- Min/Max Scaler
- Binarizer
- Bucketizer
- ...

## Categorical Features Transformers

- StringIndexer
- One-HotEncoder
- ...

## NLP Features Transformers

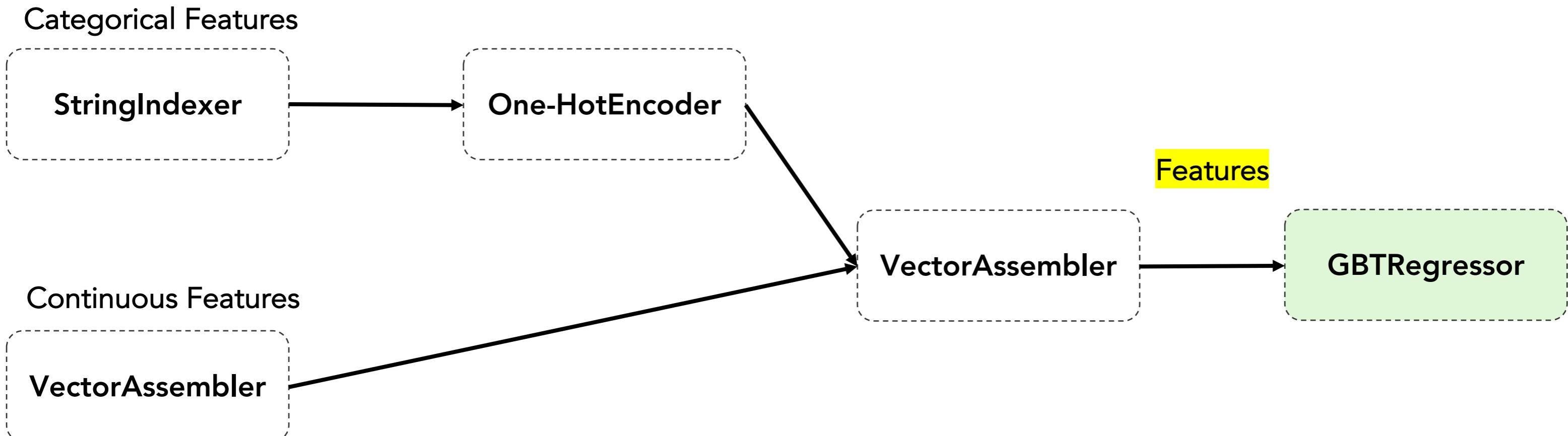
- NGram
- Word2Vec
- Hash Term Frequency
- ...

All about Feature Extraction:

<https://spark.apache.org/docs/latest/ml-features.html>

## USED SPARK PREDICTION PIPELINE

What are the steps?



Let's get out of here and see some code, please!

**REST API on Top  
of Spark ML  
Models**

# SPARK MODELS AS A REST API

## Deploying in Production

### Using Spark

- Through Spark, Streaming or JobServer (with Livy) where the prediction is performed for each micro-batch or active spark context.

### Without Spark

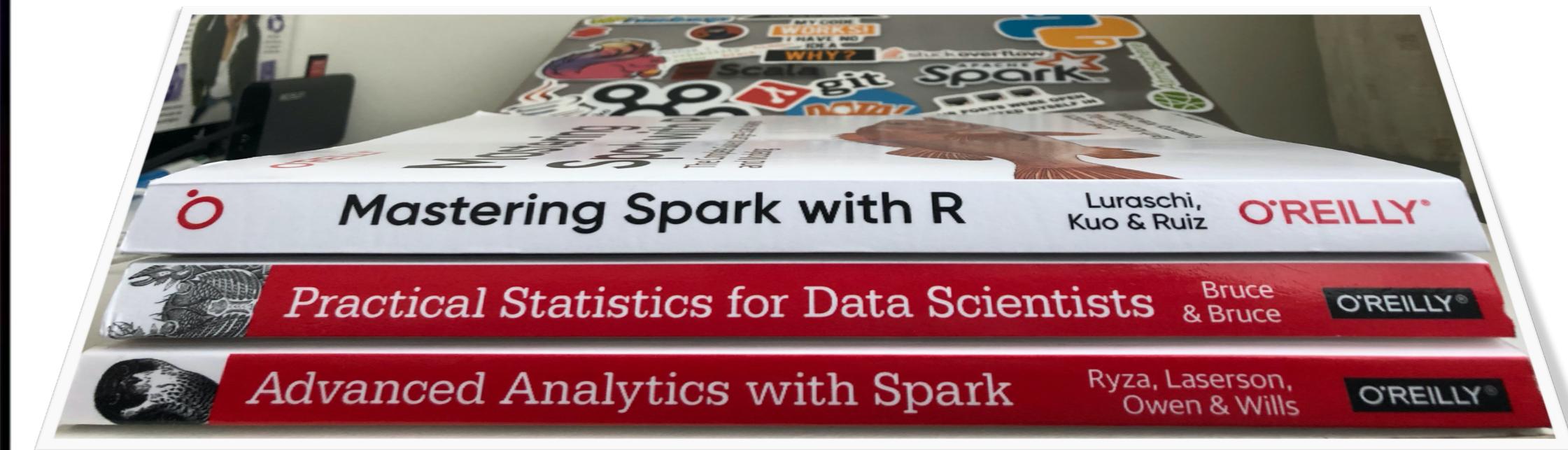
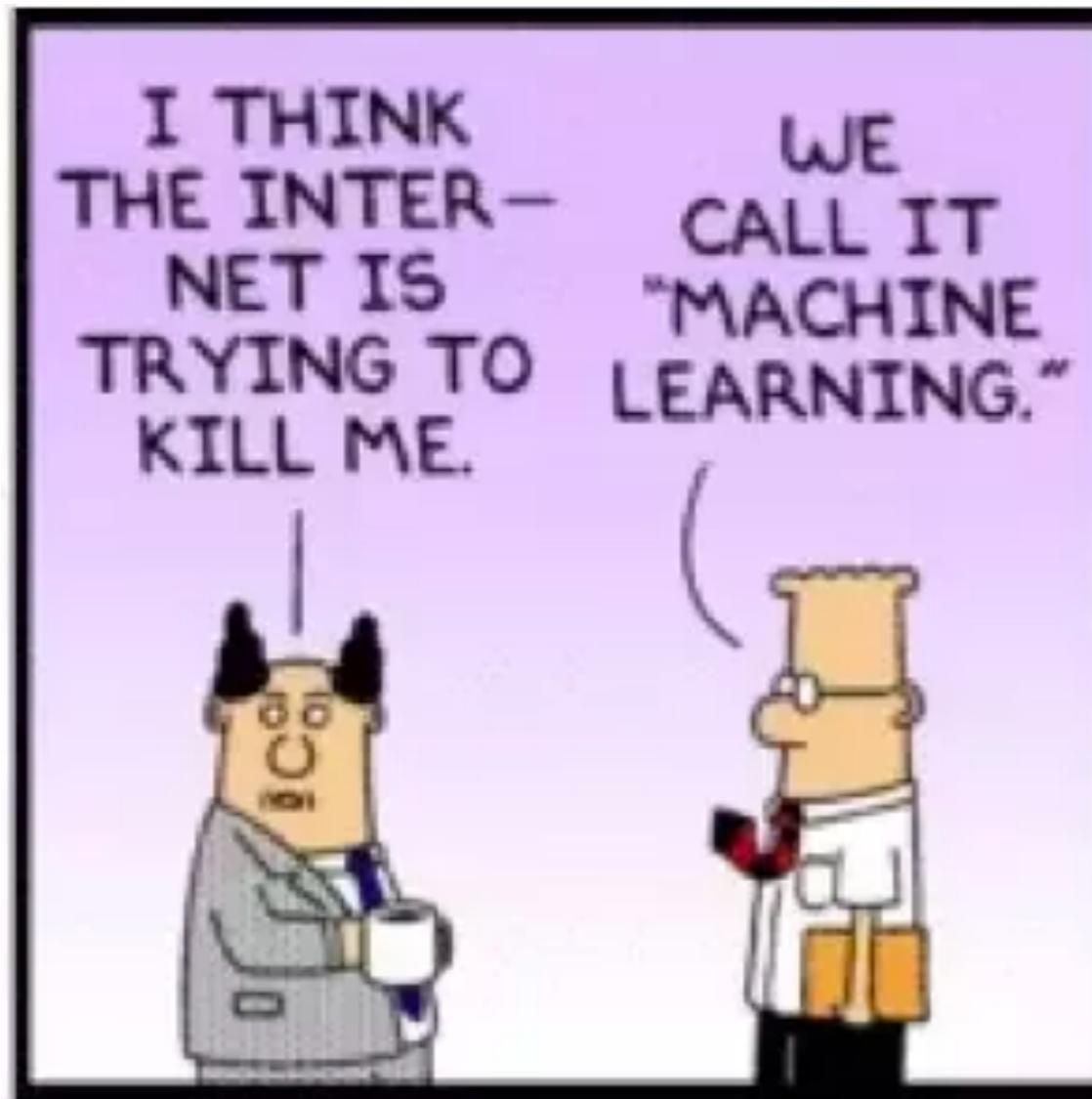
These solutions has as objective to deploy spark models to be used outside Spark without a SparkContext

- **PMML (Predictive Model Markup Language):** Supported initially in the Mllib with RDD or using a Third-Party component.
- **MLeap:** It's becoming the most applied solution and recommended also by Databricks:
  - <https://docs.databricks.com/applications/machine-learning/model-export-import/mleap-model-export.html#export-and-import-models-in-python>
  - <https://docs.databricks.com/applications/mlflow/mleap-model-deployment-on-sagemaker.html>

## REFERENCES

- A. Recommendation Systems: <https://github.com/maxhumber/BRE>
- B. Spark ALS Parameters:
  1. <https://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.ml.recommendation.ALS>
  2. <https://spark.apache.org/docs/2.4.4/ml-collaborative-filtering.html#collaborative-filtering>
- C. Business Analytics: Decision Making Using Data Course – University Cambridge Judge Business School
- D. Feature Extraction: <https://spark.apache.org/docs/2.4.4/ml-features.html>
- E. Spark PMML: <https://spark.apache.org/docs/2.3.0/mllib-pmml-model-export.html>
- F. JPMML-Evaluator-Spark: <https://github.com/jpmml/jpmml-evaluator-spark>
- G. Mleap Documentation: <https://mleap-docs.comburst.ml/>
- H. Mleap Git Repo: <https://github.com/comburst/mleap>
- I. Spark API: <https://www.codementor.io/@jadianes/building-a-web-service-with-apache-spark-flask-example-app-part2-du1083854>

# That's all folks!



Some recommended books!