



THE FUTURE IS NOW



Scalable Predictive Pipelines with Spark and Scala

Tom van der Weide

Based on a presentation from Dimitris Papadopoulos



SCHIBSTED
MEDIA GROUP



About Schibsted

The screenshot shows the homepage of the Schibsted website. At the top, there's a navigation bar with links for English, Norsk, Contact, and a search function. Below the navigation is a main menu with links for MARKETPLACES, MEDIA HOUSES, ABOUT SCHIBSTED, CAREER, INVESTOR RELATIONS, and PRESS. The main content area features a blurred background image of people in a city street. Overlaid on this is a blue banner with the text "SCHIBSTED PRODUCTS AND TECHNOLOGY" and a "Read More >" button. Below the banner, there are three small images showing people in various professional settings. On the left, there's a text block about the company's international presence. In the center, there's a graphic showing 6,800 employees across 30 countries, accompanied by a world map with blue dots indicating locations. On the right, there's a link to the Oslo Stock Exchange and a "Investor Relations >" button.

Schibsted is an international media group with

6,800 employees in **30** countries

Schibsted (SCH) is listed at the Oslo Stock Exchange
[Investor Relations >](#)

About Schibsted

We're an online classifieds company ...

We have 36+ classified sites and local marketplaces.

AND
MANY
MORE

About Schibsted

... with a global footprint ...

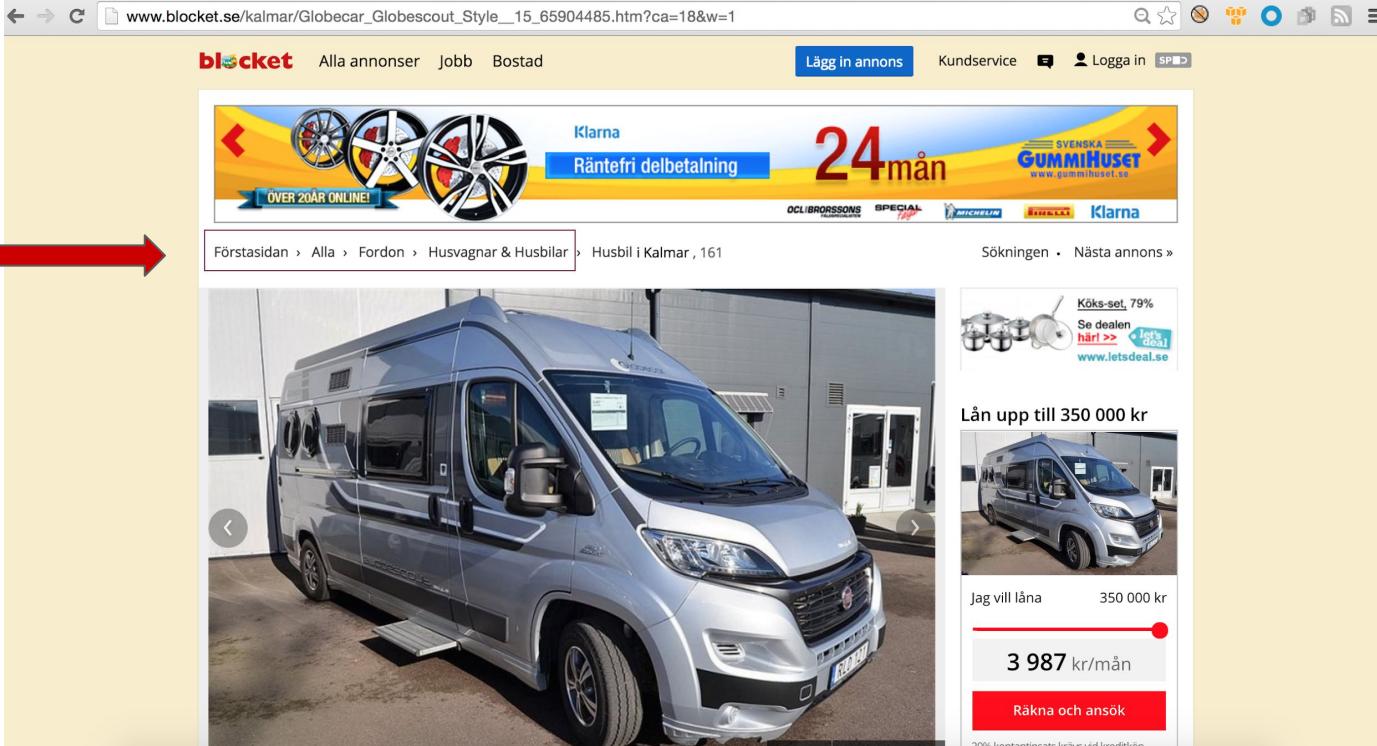
30 countries and 6800 employees. And growing rapidly.



Event Tracking Data

A screenshot of a Finn.no website page for a used Audi A4. The page shows a silver Audi A4 in a parking garage. A red arrow points to the breadcrumb navigation bar at the top left, which reads "Bil / Biler i Norge / Audi / A4". The main image of the car is displayed with navigation arrows on either side. To the right of the car, there is a user profile for "Vegard" with a blue checkmark, indicating a verified account. Below the profile, it says "Har vært på FINN i 2 år." (Has been on FINN for 2 years). There is a "Send melding" (Send message) button. Further down, the address "Flisnesvegen 52A, 6013 Ålesund" is listed, accompanied by a map. On the far right, there is an advertisement for "Gjensidige" insurance, stating "Også en Audi fra 2009 bør ha 8 års garanti på reparasjoner." (Also an Audi from 2009 should have 8 years of warranty on repairs). A call-to-action button says "Sjekk forsikring på denne bilen" (Check insurance for this car).

Event Tracking Data



A screenshot of a website showing a car advertisement. The URL in the address bar is www.blocket.se/kalmar/Globecar_Globescout_Style_15_65904485.htm?ca=18&w=1. The page header includes the blocket logo, navigation links for Alla annonser, Jobb, Bostad, and a button to Lägg in annons. The main content features a banner for Klarna Räntefri delbetalning with a 24-month offer, featuring the Svenska Gummihuset logo. Below the banner is a navigation breadcrumb: Förstasidan > Alla > Fordon > Husvagnar & Husbilar > Husbil i Kalmar , 161. To the right of the main content are search and next/previous buttons. A red arrow points to the left side of the image, highlighting the sidebar area.

Klarna
Räntefri delbetalning
24mån
SVENSKA
GUMMIHUSET
www.gummihuset.se

OCLIBRÖRSSENS SPECIAL Michelin EASYK

Förstasidan > Alla > Fordon > Husvagnar & Husbilar > Husbil i Kalmar , 161

Sökningen • Nästa annons »

Köks-set, 79%
Se dealen här >> www.letsdeal.se

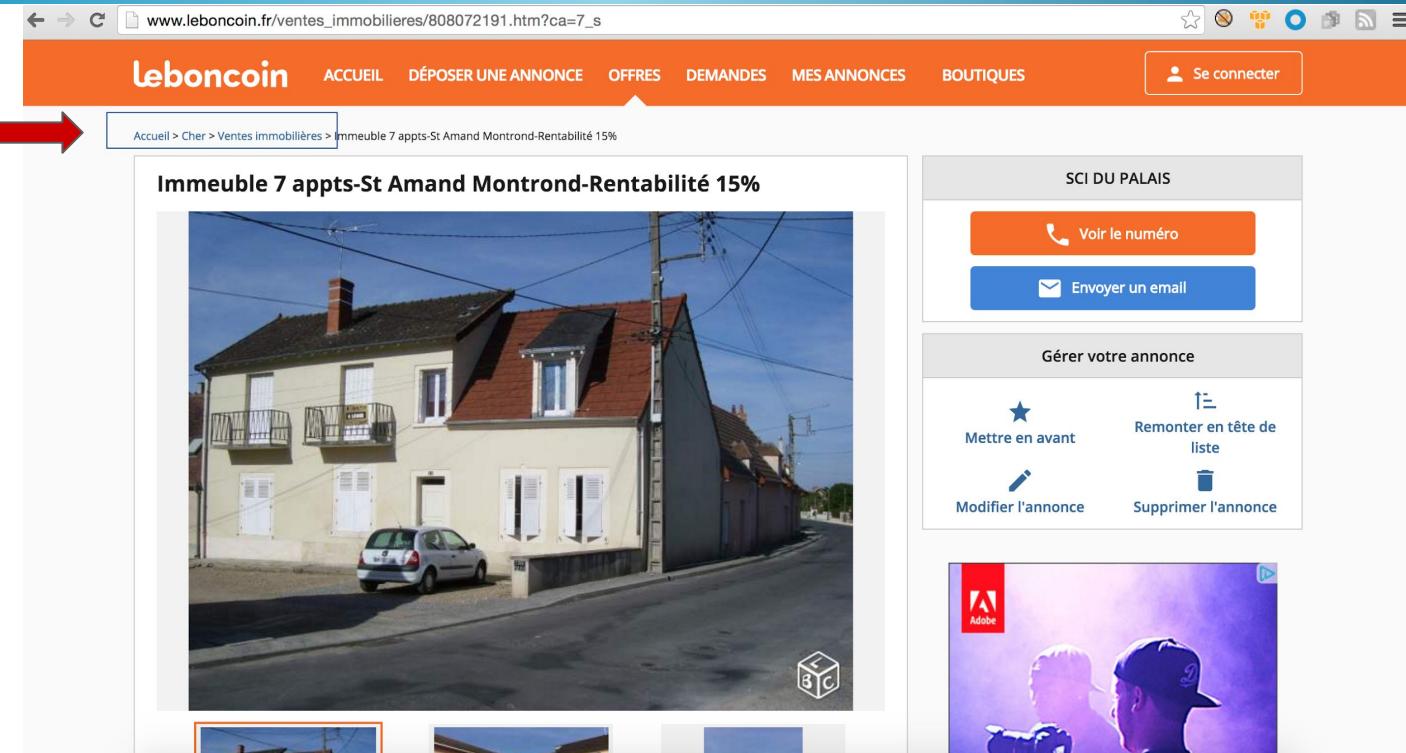
Lån upp till 350 000 kr

Jag vill låna 350 000 kr
3 987 kr/mån
Räkna och ansök

20% kontantinsats krävs vid kreditförfrågan



Event Tracking Data



www.leboncoin.fr/ventes_immobilieres/808072191.htm?ca=7_s

leboncoin ACCUEIL DÉPOSER UNE ANNONCE OFFRES DEMANDES MES ANNONCES BOUTIQUES Se connecter

Accueil > Cher > Ventes immobilières > Immeuble 7 appts-St Amand Montrond-Rentabilité 15%

Immeuble 7 appts-St Amand Montrond-Rentabilité 15%



SCI DU PALAIS

Voir le numéro Envoyer un email

Gérer votre annonce

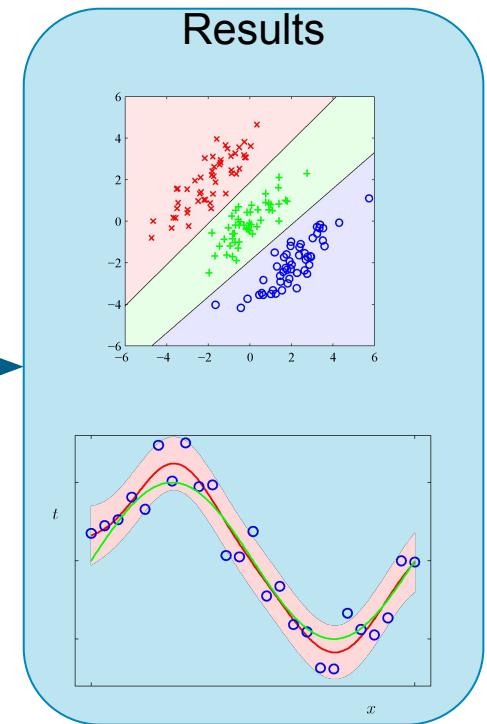
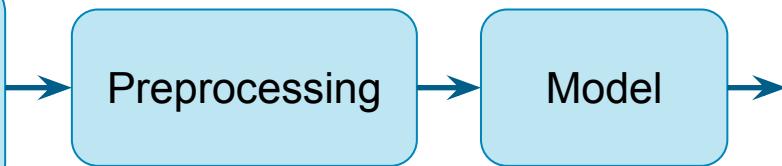
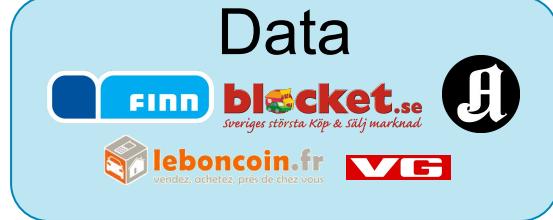
Mettre en avant Remonter en tête de liste
Modifier l'annonce Supprimer l'annonce



Event Tracking Data

The screenshot shows the Shpock homepage (en.shpock.com) with a red arrow pointing to the category navigation bar. The page features a green header with the Shpock logo and a search bar. Below the header is a grid of categories: Fashion and Accessories, Home and Garden, Electronics, Movies, Books and Music, Baby and Child, Sport, Leisure and Games, Cars and Motors, Services, and Other. To the right of the categories is a 'Sell' button and a 'Shop now' call-to-action with an arrow. Further down the page, there are sections for 'New in your area', 'YoungDesigner', and various seasonal categories like 'Giveaways', 'Friends & Following', 'Spring • Outdoor', and 'Spring • Fashion'. A large image of a woman's face with red lips and green nail polish is prominently displayed. Below this image are download links for the App Store and Google Play. The bottom of the page includes a sidebar for 'MO DEDICA' and a footer with a location search bar and other navigation links.

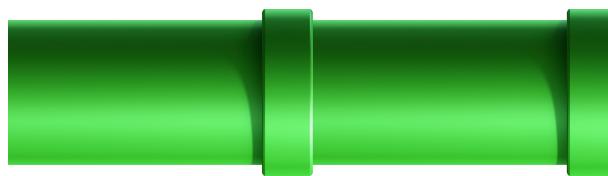
Data Science Tasks



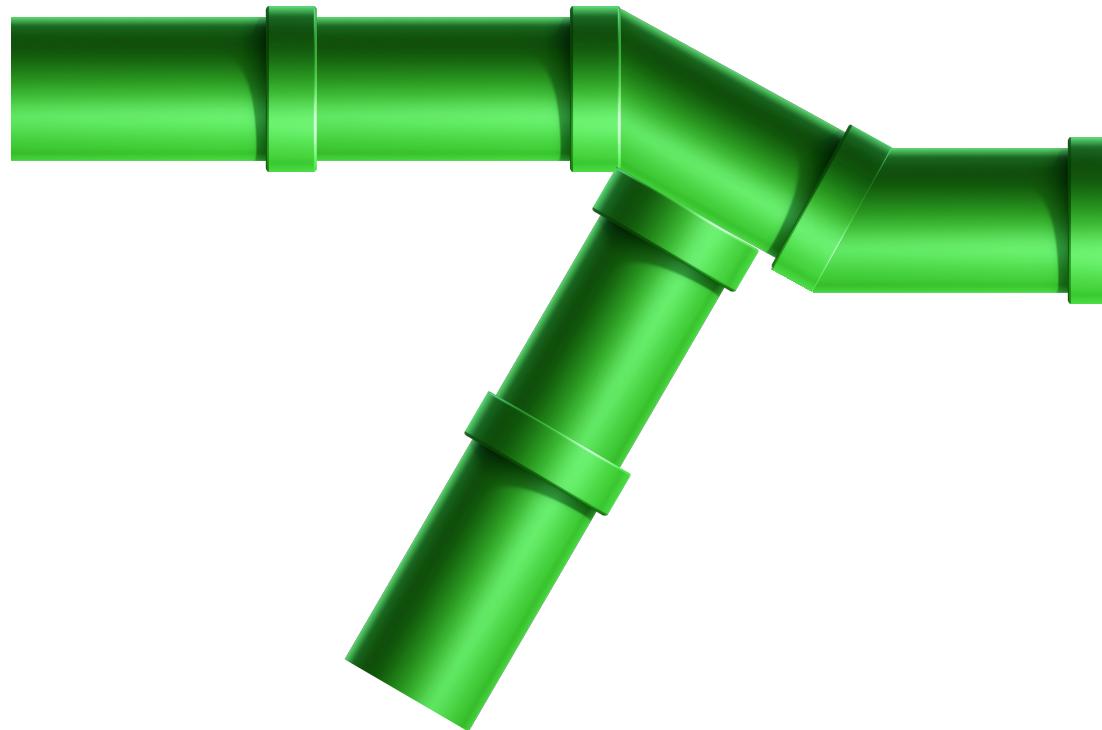
Outline

1. Using Spark ML Pipelines
2. Scalable Pipelines

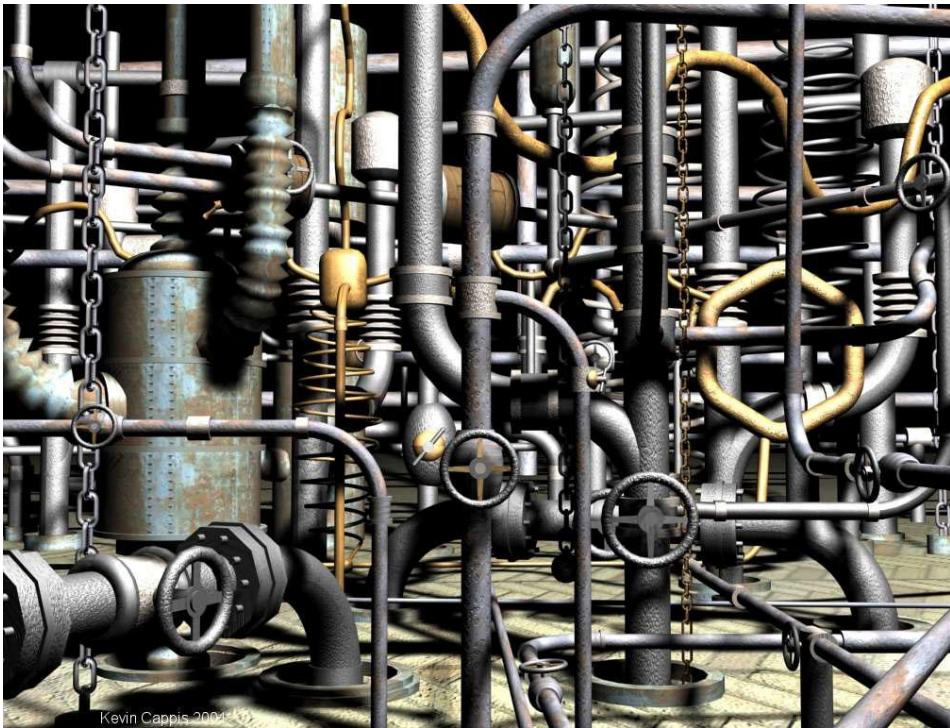
Pipeline



Pipeline



Pipeline



Kevin Cappis 2004

Pipeline



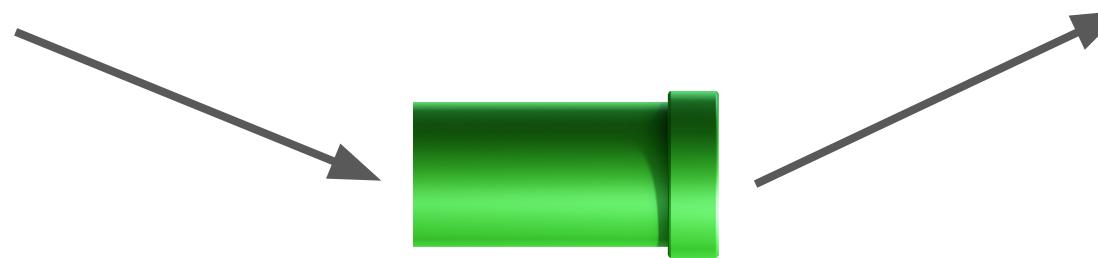
Check out:
Machine Learning: The High-Interest Credit Card of Technical Debt
by Google employees

Not a pipe



Pipeline Stage

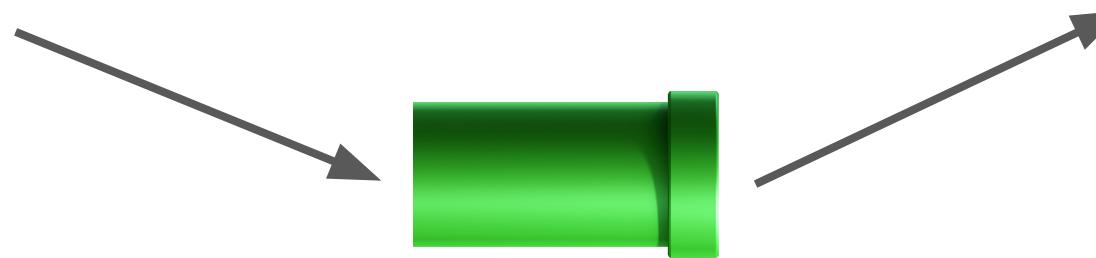
One or more inputs



Strictly one output

Pipeline Stage

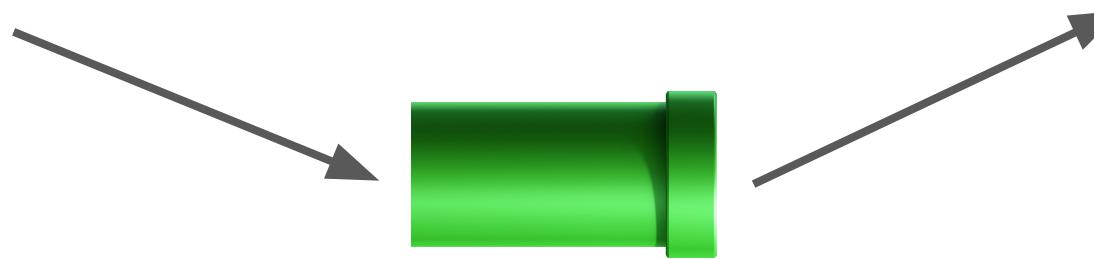
One or more inputs



- Closed under concatenation

Pipeline Stage

One or more inputs



- Closed under concatenation
- Standalone and runnable
- Spark™ ML inside

Spark ML Pipelines

The screenshot shows a navigation bar at the top with the Spark logo (1.6.1), followed by links for Overview, Programming Guides, API Docs, Deploying, and More. A sidebar on the left lists various machine learning concepts under the heading "transformation". The main content area features a section titled "Main concepts in Pipelines" with a list of five bullet points explaining DataFrame, Transformer, Estimator, Pipeline, and Parameter.

transformation

- Frequent pattern mining
- Evaluation metrics
- PMML model export
- Optimization (developer)

◦ Example: model selection via train validation split

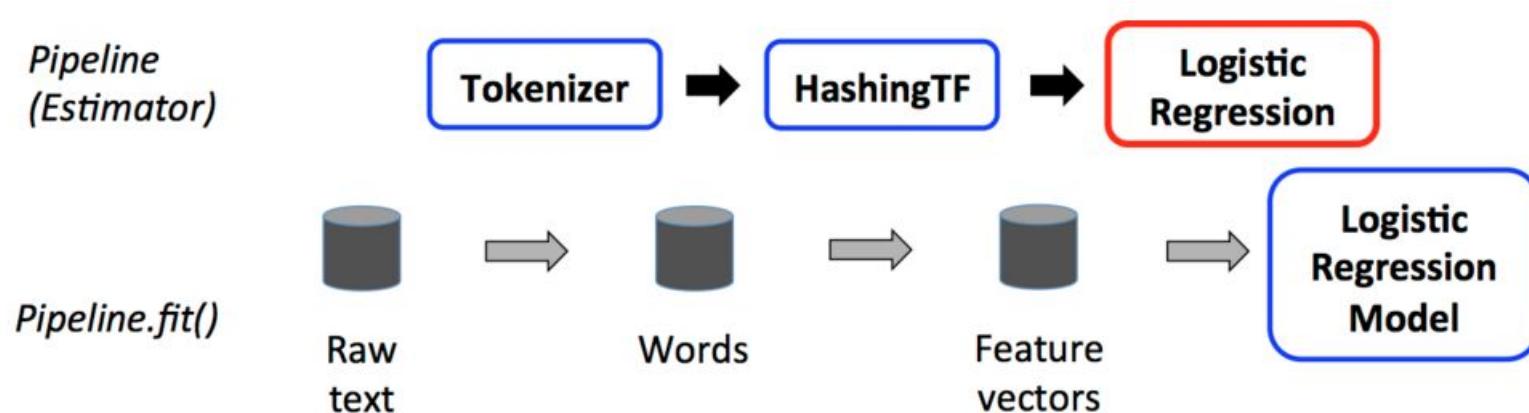
Main concepts in Pipelines

Spark ML standardizes APIs for machine learning algorithms to make it easier to combine multiple algorithms into a single pipeline, or workflow. This section covers the key concepts introduced by the Spark ML API, where the pipeline concept is mostly inspired by the [scikit-learn](#) project.

- **DataFrame**: Spark ML uses DataFrame from Spark SQL as an ML dataset, which can hold a variety of data types. E.g., a DataFrame could have different columns storing text, feature vectors, true labels, and predictions.
- **Transformer**: A Transformer is an algorithm which can transform one DataFrame into another DataFrame. E.g., an ML model is a Transformer which transforms DataFrame with features into a DataFrame with predictions.
- **Estimator**: An Estimator is an algorithm which can be fit on a DataFrame to produce a Transformer. E.g., a learning algorithm is an Estimator which trains on a DataFrame and produces a model.
- **Pipeline**: A Pipeline chains multiple Transformers and Estimators together to specify an ML workflow.
- **Parameter**: All Transformers and Estimators now share a common API for specifying parameters.

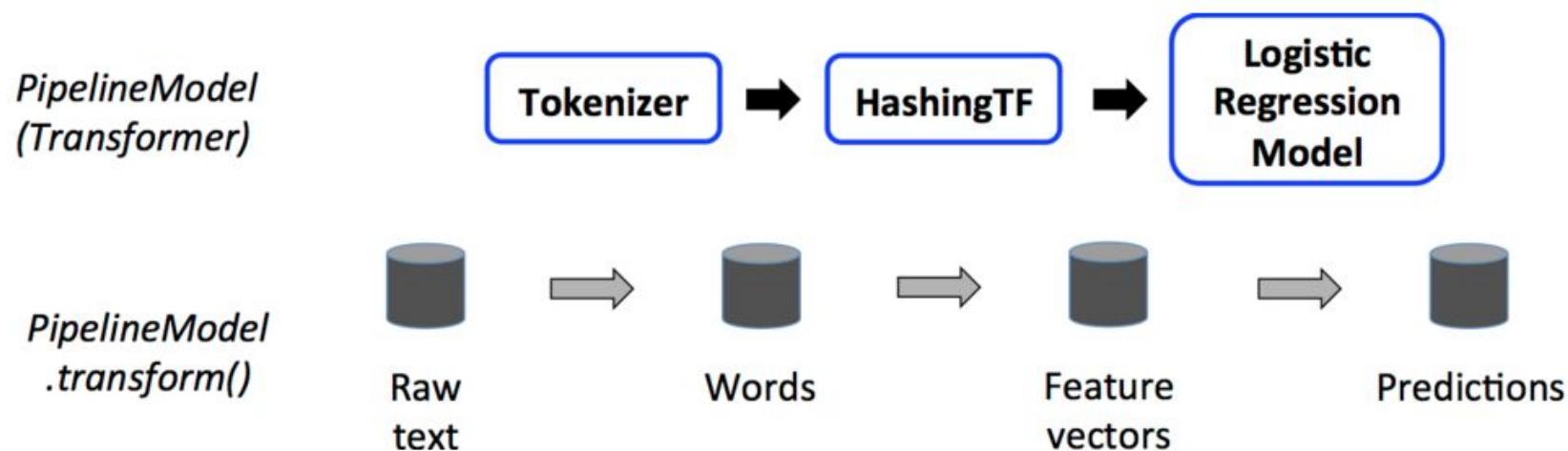
Spark ML Pipelines

Using a Pipeline to train a model

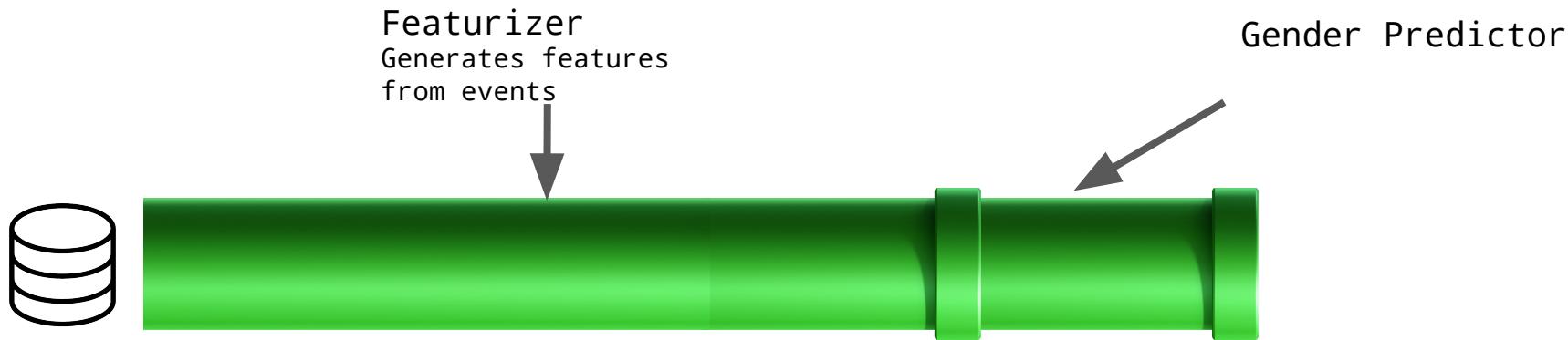


Spark ML Pipelines

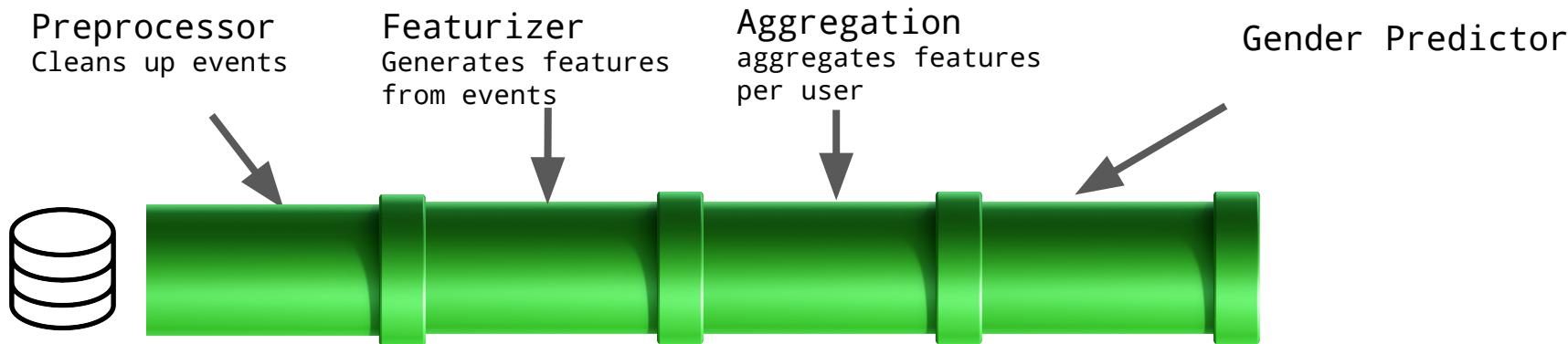
Using a PipelineModel to get predictions



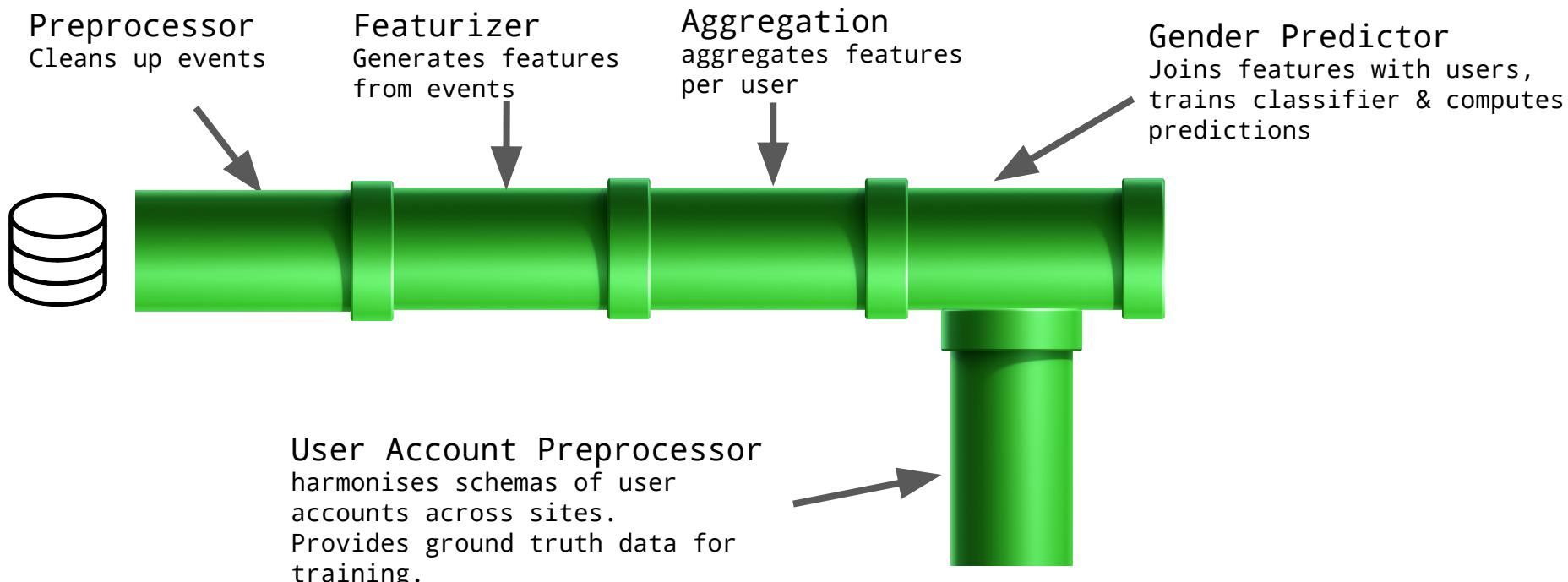
Example: Gender Prediction Pipeline



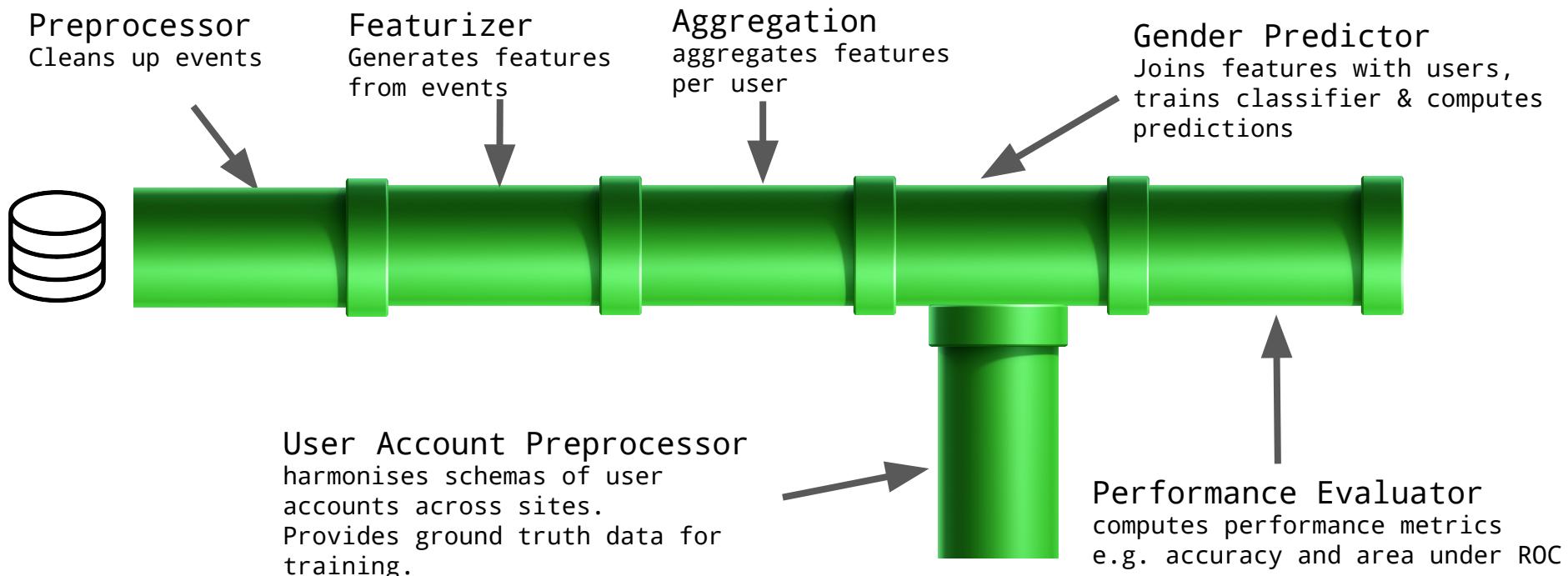
Example: Gender Prediction Pipeline



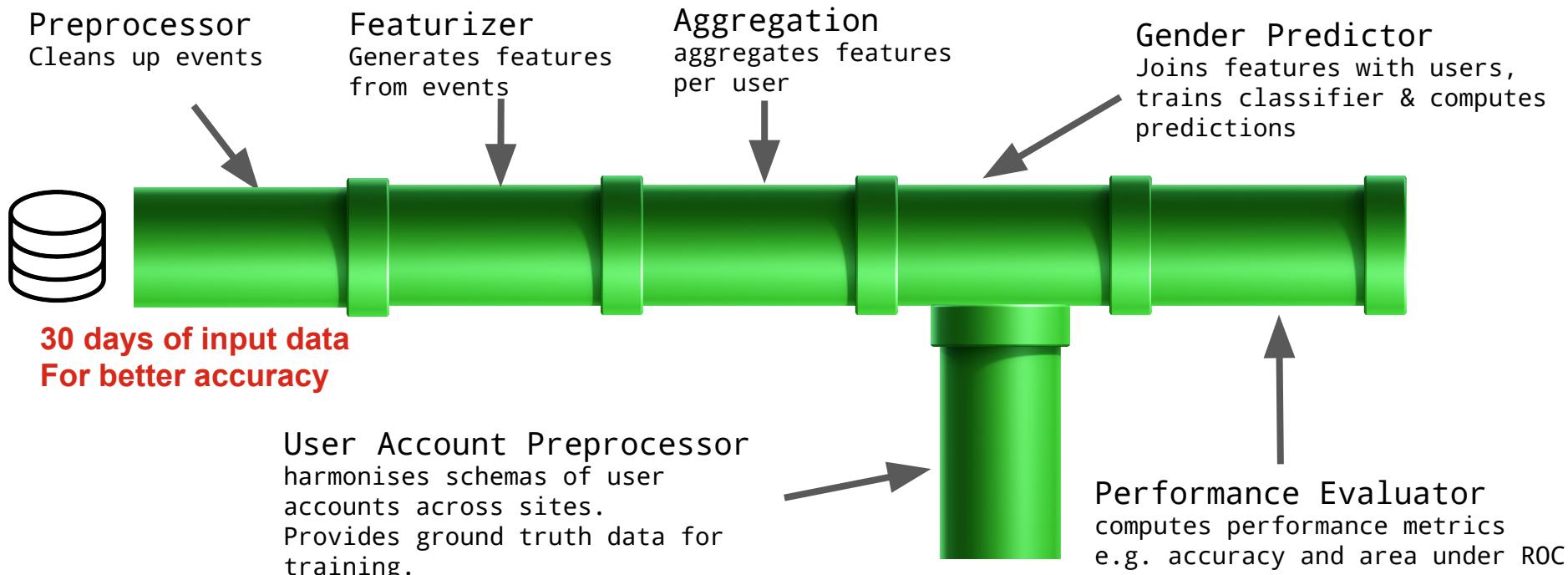
Example: Gender Prediction Pipeline



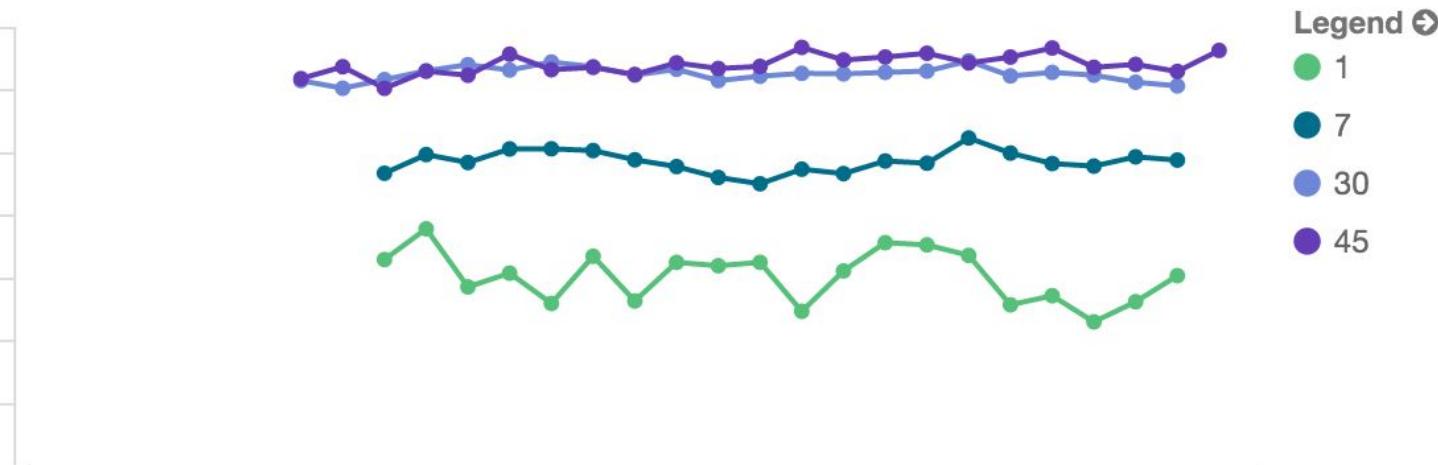
Example: Gender Prediction Pipeline



Scalability pain points

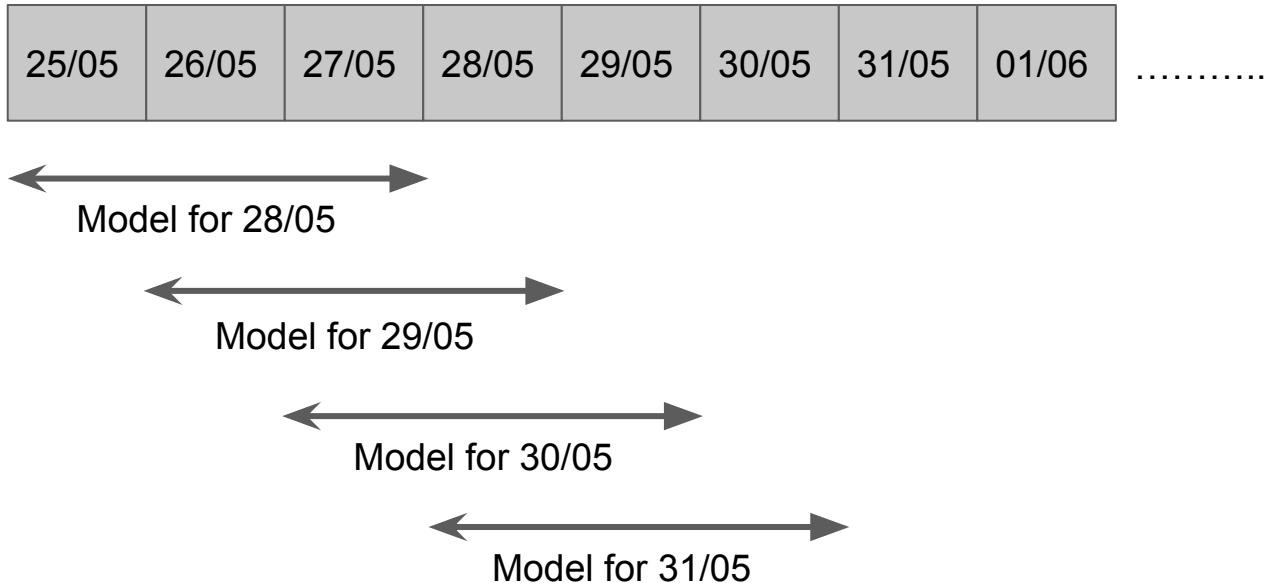


More data for better performance



Redundant processing

Input data per day



Scalable Pipelines: pain points

*“What will happen if we try to process
30 days worth of data (e.g. 6B events) ???”*

Memory and processing heavy

- In one use-case, for 7 days lookback (~7 x 200M events)
we used to need 20 Spark executors with 22G of memory each.

Not easily scalable

- As the lookback increases
- As more and more sites are incorporated into our pipelines

Redundant processing

- For K days of lookback, we are repeating processing of K-1 days worth of data,
when we run the pipeline every day, in a rolling window fashion.

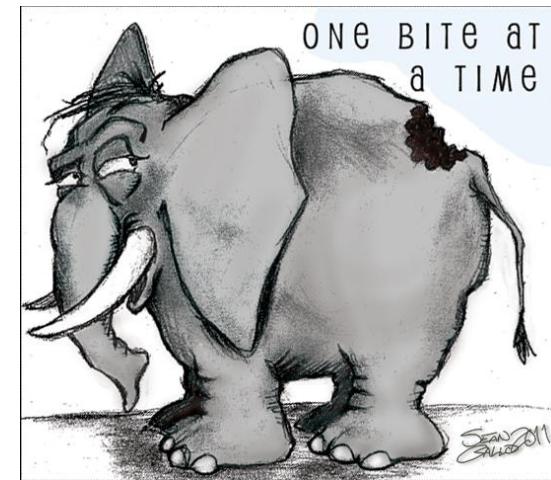
Saved by Algebra

- The operations (op) along with the corresponding data structures (S) that we are interested in are ***monoids***.
 - Associative:
 - for all A, B, C in S , $(A \ op \ B) \ op \ C = A \ op \ (B \ op \ C)$
 - Identity element:
 - there exists E in S such that for each A in S , $E \ op \ A = A \ op \ E = A$
- Examples:
 - Summation: $1 + 2 + 3 + 4 = (1 + 2) + (3 + 4) = 1 + ((2 + 3) + 4)$
 - String array concatenation: $["foo"] + ["bar"] + ["baz"] = ["foo", "bar"] + ["baz"]$

Scalable Pipelines: in monoids fashion

Split work into smaller chunks, cache intermediate results

- Split the aggregations into smaller chunks
 - Persist aggregations per day
- Combine partial aggregations into final feature vectors



Scalable Pipelines: building blocks

User Defined Functions (UDF)

compute a value looking at a single row

e.g. `SELECT concat('user', user_id) FROM events`

User Defined Aggregate Functions (UDAF) since Spark 1.5

computation over groups of rows

e.g. `SELECT user_id, AVG(time_spent) FROM events GROUP BY user_id`

Example: map with term frequencies

For some user, we already aggregated events per day:

```
val day1 = Map("cars" -> 2.0, "kitchen" -> 1.0, "house" -> 3.0)  
val day2 = Map("kitchen" -> 4.0, "house" -> 1.0, "furniture" -> 5.0)
```

Monoid operation: MapAggregator \oplus

i.e. $m1 \oplus m2 \oplus m3 = (m1 \oplus m2) \oplus m3 = m1 \oplus (m2 \oplus m3)$

$m0 = \text{Map}()$

Result:

```
val together = Map("cars" -> 2.0, "kitchen" -> 5.0, "house" -> 4.0, "furniture" -> 5.0)
```

```
object MapAggregator extends UserDefinedAggregateFunction {
    type K = String
    type V = Double

    override def initialize(buffer: MutableAggregationBuffer): Unit =
        buffer.update(0, new HashMap[K, V])

    override def update(buffer: MutableAggregationBuffer, input: Row): Unit = {
        if (!input.isNullAt(0)) {
            buffer.update(0, sum(evaluate(buffer), evaluate(input)))
        }
    }

    override def merge(buffer1: MutableAggregationBuffer, buffer2: Row): Unit =
        buffer1.update(0, sum(evaluate(buffer1), evaluate(buffer2)))

    override def evaluate(buffer: Row): Map[K, V] = ←
        buffer.getAs[Map[K, V]](0) ←
            Get content from buffer

    private def sum(map1: Map[K, V], map2: Map[K, V]): Map[K, V] =
        map1 ++ (for ((k, v) <- map2) yield k -> (v + map1.getOrDefault(k, 0.0)))

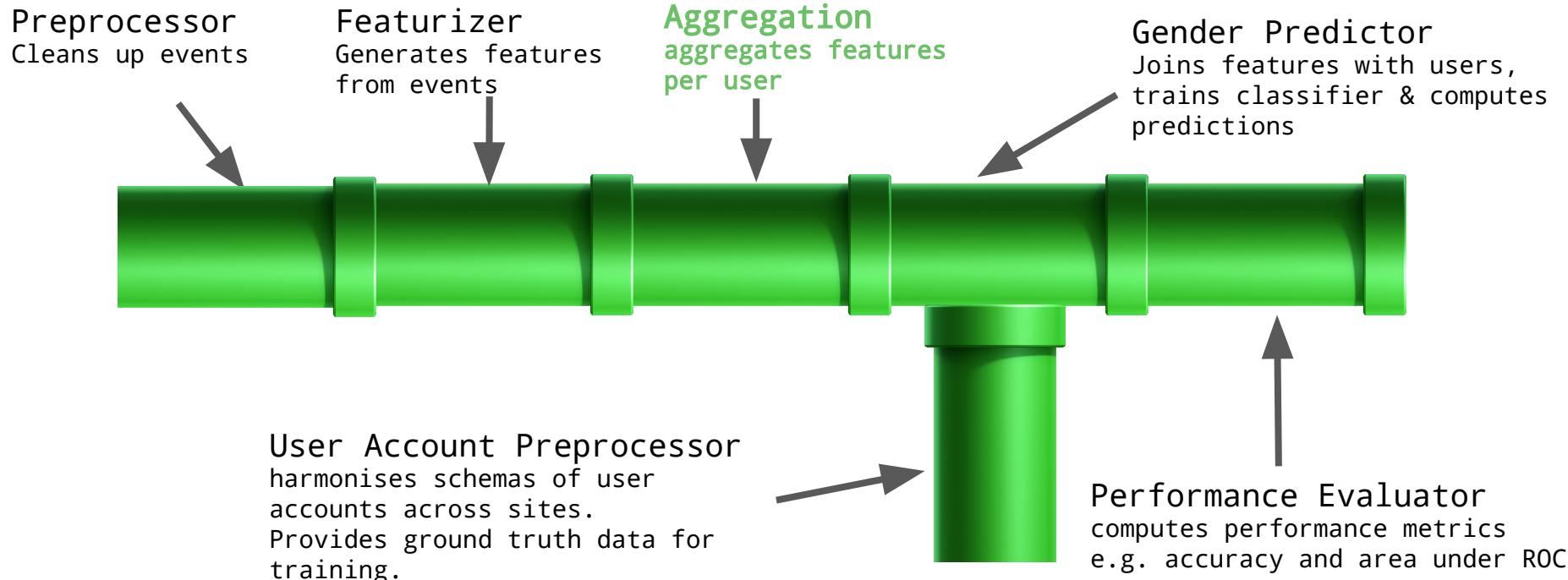
    override def dataType = DataTypes.createMapType(StringType, DoubleType, true)
    override def inputSchema = new StructType().add("input", dataType)
    override def bufferSchema = new StructType().add("input", dataType)
    override def deterministic = true
}
```

Buffer for intermediate results

Merge buffers of two executors

Get content from buffer

Aggregation pipeline stage



Scalable Pipelines: Aggregating Events

```
class EventAggregator(override val uid: String) extends Transformer {

  def this() = this(Identifiable.randomUUID("eventAggregator"))

  override def transform(dataset: DataFrame): DataFrame = {
    val aggregates = dataset.schema.fields
    .map { field: StructField => field.dataType match {
      case ArrayType(StringType, _) => concat_array_string(col(field.name)).as(field.name)
      case IntegerType | LongType | ShortType | FloatType | DoubleType => sum(col(field.name)).as(field.name)
      case VectorType => SparseVectorAggregator(col(field.name)).as(field.name)
      case MapType(StringType, DoubleType, _) => MapAggregator(col(field.name)).as(field.name)
      case _ => col(field.name).as(field.name)
    }}
    dataset.groupBy(col(userId), col(isAnonymous))
      .agg(aggregates.head, aggregates.tail: _*)
  }

  override def transformSchema(schema: StructType): StructType = schema

  override def copy(extra: ParamMap): Transformer = defaultCopy(extra)
}
```

Scalable Pipelines: Aggregating Events

```
class EventAggregator(override val uid: String) extends Transformer { ←  
  def this() = this(Identifiable.randomUUID("eventAggregator"))  
  
  override def transform(dataset: DataFrame): DataFrame = {  
    val aggregates = dataset.schema.fields  
    .map { field: StructField => field.dataType match {  
      case ArrayType(StringType, _) => concat_array_string(col(field.name)).as(field.name)  
      case IntegerType | LongType | ShortType | FloatType | DoubleType => sum(col(field.name)).as(field.name)  
      case VectorType => SparseVectorAggregator(col(field.name)).as(field.name)  
      case MapType(StringType, DoubleType, _) => MapAggregator(col(field.name)).as(field.name)  
      case _ => col(field.name).as(field.name)  
    }  
  }  
  
  dataset.groupBy(col(userId), col(isAnonymous))  
    .agg(aggregates.head, aggregates.tail: _*)  
}  
  
override def transformSchema(schema: StructType): StructType = schema  
  
override def copy(extra: ParamMap): Transformer = defaultCopy(extra)  
}
```

It's a Transformer

Scalable Pipelines: Aggregating Events

```
class EventAggregator(override val uid: String) extends Transformer { ← It's a Transformer
  def this() = this(Identifiable.randomUUID("eventAggregator"))

  override def transform(dataset: DataFrame): DataFrame = { ← DataFrame in , DataFrame out
    val aggregates = dataset.schema.fields
    .map { field: StructField => field.dataType match {
      case ArrayType(StringType, _) => concat_array_string(col(field.name)).as(field.name)
      case IntegerType | LongType | ShortType | FloatType | DoubleType => sum(col(field.name)).as(field.name)
      case VectorType => SparseVectorAggregator(col(field.name)).as(field.name)
      case MapType(StringType, DoubleType, _) => MapAggregator(col(field.name)).as(field.name)
      case _ => col(field.name).as(field.name)
    }}
    dataset.groupBy(col(userId), col(isAnonymous))
      .agg(aggregates.head, aggregates.tail: _*)
  }

  override def transformSchema(schema: StructType): StructType = schema

  override def copy(extra: ParamMap): Transformer = defaultCopy(extra)
}
```

Scalable Pipelines: Aggregating Events

```
class EventAggregator(override val uid: String) extends Transformer { ← It's a Transformer
  def this() = this(Identifiable.randomUUID("eventAggregator"))

  override def transform(dataset: DataFrame): DataFrame = { ← DataFrame in , DataFrame out
    val aggregates = dataset.schema.fields
    .map { field: StructField => field.dataType match {
      case ArrayType(StringType, _) => concat_array_string(col(field.name)).as(field.name)
      case IntegerType | LongType | ShortType | FloatType | DoubleType => sum(col(field.name)).as(field.name)
      case VectorType => SparseVectorAggregator(col(field.name)).as(field.name)
      case MapType(StringType, DoubleType, _) => MapAggregator(col(field.name)).as(field.name)
      case _ => col(field.name).as(field.name)
    }}
    dataset.groupBy(col(userId), col(isAnonymous))
      .agg(aggregates.head, aggregates.tail: _*)
  }

  override def transformSchema(schema: StructType): StructType = schema

  override def copy(extra: ParamMap): Transformer = defaultCopy(extra)
}
```

Aggregating maps
of feature frequency
counts

Scalable Pipelines: closing remarks

- With User Defined Aggregate Functions, we have reduced the workload of our pipelines by a factor of 20!

Scalable Pipelines: closing remarks

- With User Defined Aggregate Functions, we have reduced the workload of our pipelines by a factor of 20!
- Obvious gains: freeing up resources that can be used for running even more pipelines, faster, over even more input data

Scalable Pipelines: closing remarks

- Needless to say, more factors contribute towards a scalable pipeline:
 - Performance tuning of the Spark cluster
 - Use of a workflow manager (e.g. Luigi) for pipeline orchestration

Scalable Pipelines: closing remarks

- Needless to say, more factors contribute towards a scalable pipeline:
 - Performance tuning of the Spark cluster
 - Use of a workflow manager (e.g. Luigi) for pipeline orchestration
- But each one of these is a topic for a separate talk

Shameless plug

We are hiring!

Across all our hubs

in London, Oslo, Stockholm, Barcelona

for Data Science, Engineering, UX and Product roles

<https://jobs.lever.co/schibsted>
spt-recruiters@schibsted.com

Q/A

Thank you!