Built with public APIs*

Extending Spark ML

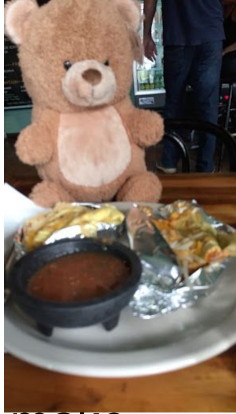
Super Happy New Pipeline Stage Time!

*Scala only - see developer for details.

Who am I?

- My name is Holden Karau
- Prefered pronouns are she/her
- I’m a Principal Software Engineer at IBM’s Spark Technology Center
- previously Alpine, Databricks, Google, Foursquare & Amazon
- co-author of Learning Spark & Fast Data processing with Spark
  - co-author of a new book focused on Spark performance coming this year*
- @holdenkarau
- Slide share http://www.slideshare.net/hkarau
- Linkedin https://www.linkedin.com/in/holdenkarau
- Github https://github.com/holdenk
- Spark Videos http://bit.ly/holdenSparkVideos

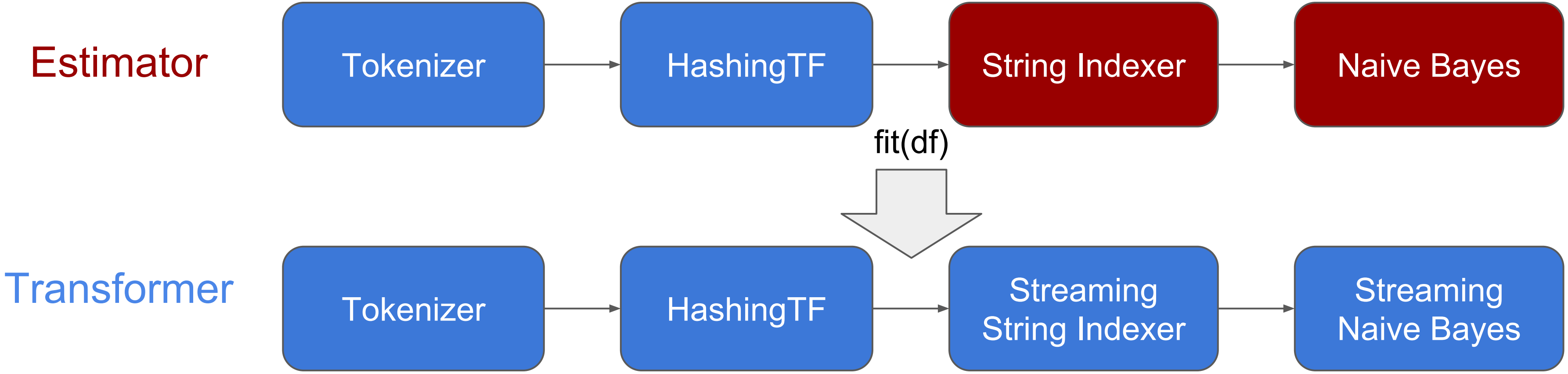
What are we going to talk about?



- What Spark ML pipelines look like
- What Estimators and Transformers are
- How to implement a Transformer - and what else you will need to do to make an estimator
- I will of course try and sell you many copies of my new book if you have an expense account.

Spark ML pipelines

- In the batch setting, an estimator is trained on a dataset, and produces a static, immutable transformer.



So what does a pipeline stage look like?

Are either an:

- Estimator - no need to train can directly transform (e.g. HashingTF) (with transform)
- Transformer - has a method called “fit” which returns an estimator

Must provide:

- transformSchema (used to validate input schema is reasonable) & copy

Often have:

- Special params for configuration (so we can do meta-algorithms)

Walking through a simple transformer:


```scala
class HardCodedWordCountStage(override val uid: String) extends
Transformer {
  def this() = this(Identifiable.randomUID("hardcodedwordcount"))

  def copy(extra: ParamMap): HardCodedWordCountStage = {
    defaultCopy(extra)
  }
```

Verify the input schema is reasonable:

```scala
override def transformSchema(schema: StructType): StructType = {
  // Check that the input type is a string
  val idx = schema.fieldIndex("happy_pandas")
  val field = schema.fields(idx)
  if (field.dataType != StringType) {
    throw new Exception(s"Input type ${field.dataType} did not match input type StringType")
  }
  // Add the return field
  schema.add(StructField("happy_panda_counts", IntegerType, false))
}
```

Do the “work” (e.g. predict labels or w/e):


```
def transform(df: Dataset[_]): DataFrame = {
    val wordcount = udf { in: String => in.split(" ").size }
    df.select(col("*"),
      wordcount(df.col("happy_pandas")).as("happy_panda_counts"))
  }
```

What about configuring our stage?


```scala
class ConfigurableWordCount(override val uid: String) extends
Transformer {
  final val inputCol= new Param[String](this, "inputCol", "The input
column")
  final val outputCol = new Param[String](this, "outputCol", "The
output column")

  def setInputCol(value: String): this.type = set(inputCol, value)

  def setOutputCol(value: String): this.type = set(outputCol, value)
```

So why do we configure it that way?

- Allow meta algorithms to work on it
- If you like inside of spark you’ll see “sharedParams” for common params (like input column)
- We can access those unless we pretend to be inside of org.apache.spark - so we have to make our own

So how to make an estimator?

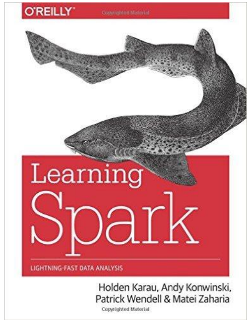
- Very similar, instead of directly providing transform provide a `fit` which returns a “model” which implements the estimator interface as shown above
- We could look at one - but I’m only supposed to talk for 10 minutes
- So keep an eye out for my blog post in November :)
- Also take a look at the algorithms in Spark itself (helpful traits you can mixin to take care of many common things).

Resources to continue with:

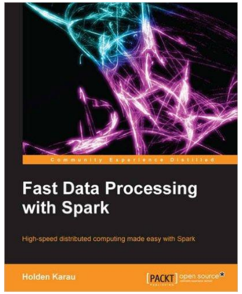
- O'Reilly Radar ("Ideas") Blog Post

  http://bit.ly/extendSparkML

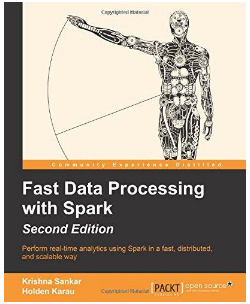
- High Performance Spark Example Repo has some sample "custom" models https://github.com/high-performance-spark/high-performance-spark-examples
  - Of course buy several copies of the book - it is the gift of the season :p
- The models inside of Spark its self: https://github.com/apache/spark/tree/master/mllib/src/main/scala/org/apache/spark/ml (use some internal APIs but a good starting point)
- As always the Spark API documentation: http://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.package
- My Slide share http://www.slideshare.net/hkarau



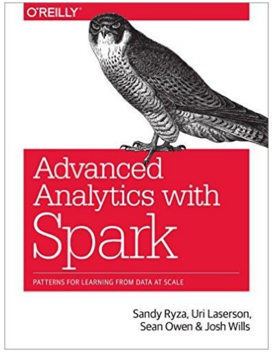
Learning Spark



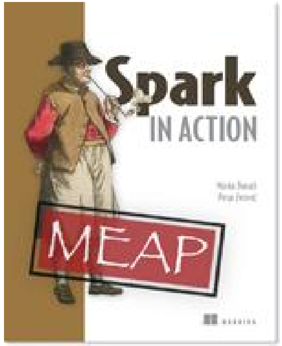
Fast Data
Processing with
Spark
(Out of Date)



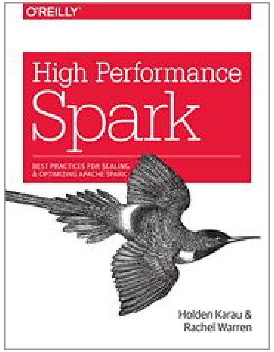
Fast Data
Processing with
Spark
(2nd edition)



Advanced
Analytics with
Spark

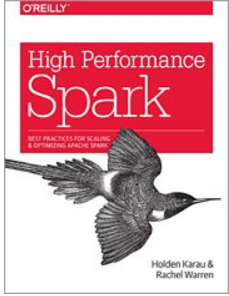


Coming soon:
Spark in Action



Coming soon:
High Performance Spark

The next book…..



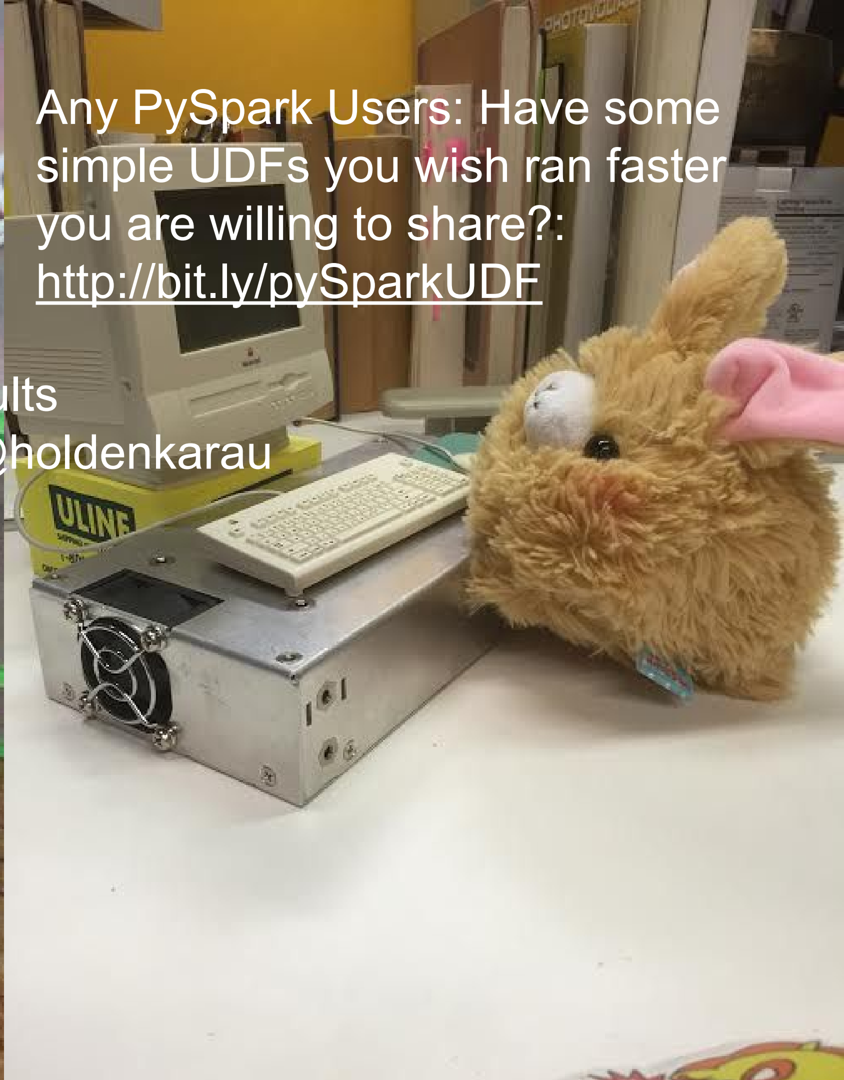
First seven chapters are available in “Early Release”*:

- Buy from O’Reilly - http://bit.ly/highPerfSpark
- Extending ML is covered in Chapter 9 :)

Get notified when updated & finished:

- http://www.highperformancespark.com
- https://twitter.com/highperfspark

* Early Release means extra mistakes, but also a chance to help us make a more awesome book.



k thnx bye :)

Any PySpark Users: Have some simple UDFs you wish ran faster you are willing to share?: http://bit.ly/pySparkUDF

Will tweet results "eventually" @holdenkarau

If you care about Spark testing and don't hate surveys: http://bit.ly/holdenTestingSpark

Pssst: Have feedback on the presentation? Give me a shout (holden@pigscanfly.ca) if you feel comfortable doing so :)