Spark: Data Science as a Service

Sridhar Alla, Kiran Muglurmath

Comcast

Who we are

- Sridhar Alla

  Director, Data Engineering, Comcast

  Architecting and building solutions on a big data scale

  sridhar_alla@cable.comcast.com

- Kiran Muglurmath

  Executive Director, Data Science, Comcast

  Data science on a big data scale.

  kiran_muglurmath@cable.comcast.com

Agenda

- Why do this?
- Where are we now
- Real world challenges
- Introduction to Roadrunner – Our Solution to the real world challenges
- How we use Roadrunner in Comcast
- Q & A

Our Data

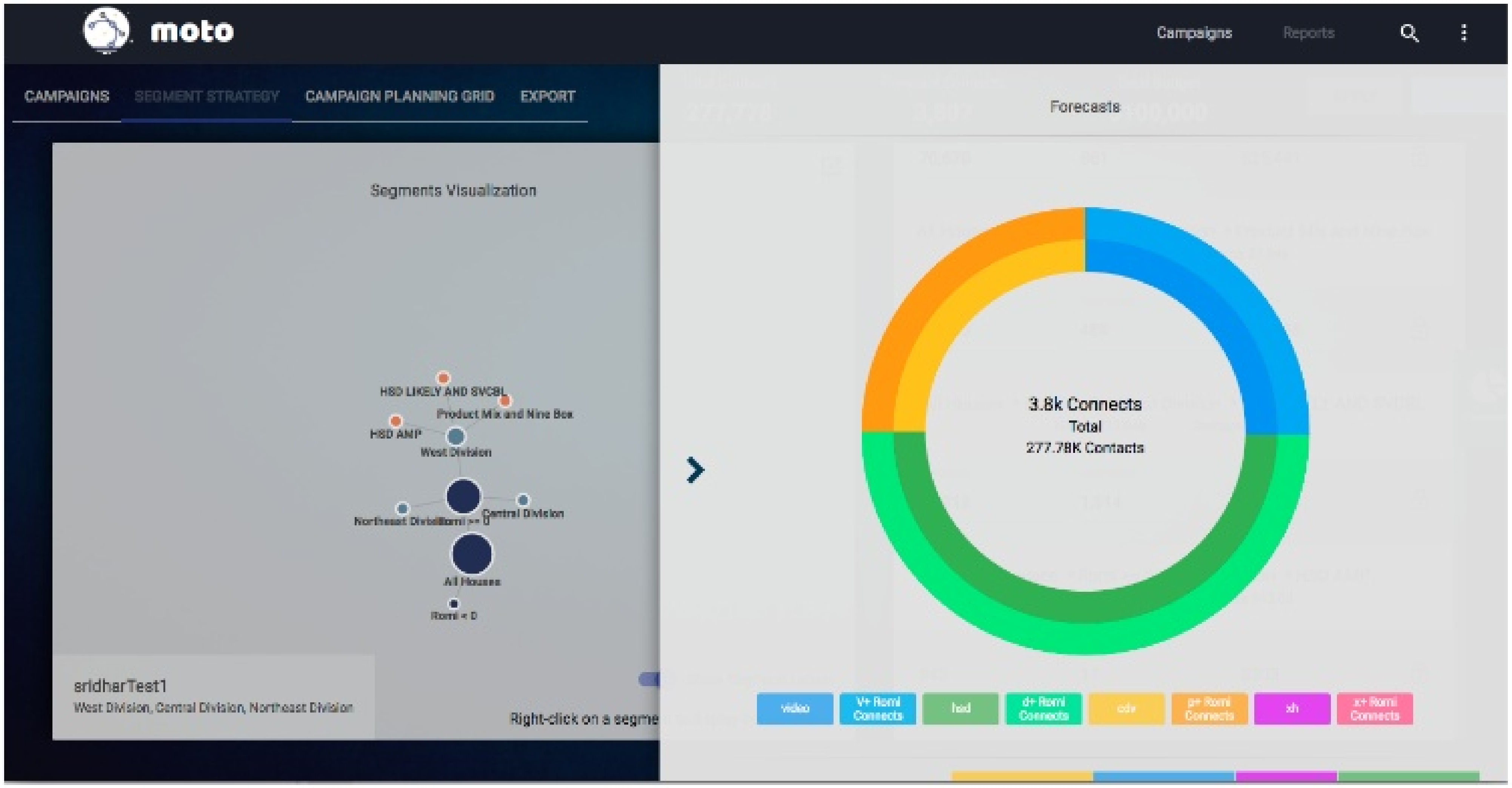
- 40PB in HDFS capacity and 100s of TBs in Teradata space
- ~1200 data nodes in total in Hadoop and Spark clusters
- Multiple 1Trillion+ row datasets
- Datasets with 12000+ columns
- 100s of models
  - Logistic regression, Neural Networks
  - LDA and other text analytics
  - Bayesian Networks
  - Clustering that includes kmeans, hierarchical, density
  - Geospatial

Data Science Use Cases

- Churn Models
- Price Elasticity
- Geo Spatial Route Optimization
- Direct Mail Campaign
- Customer call Analytics

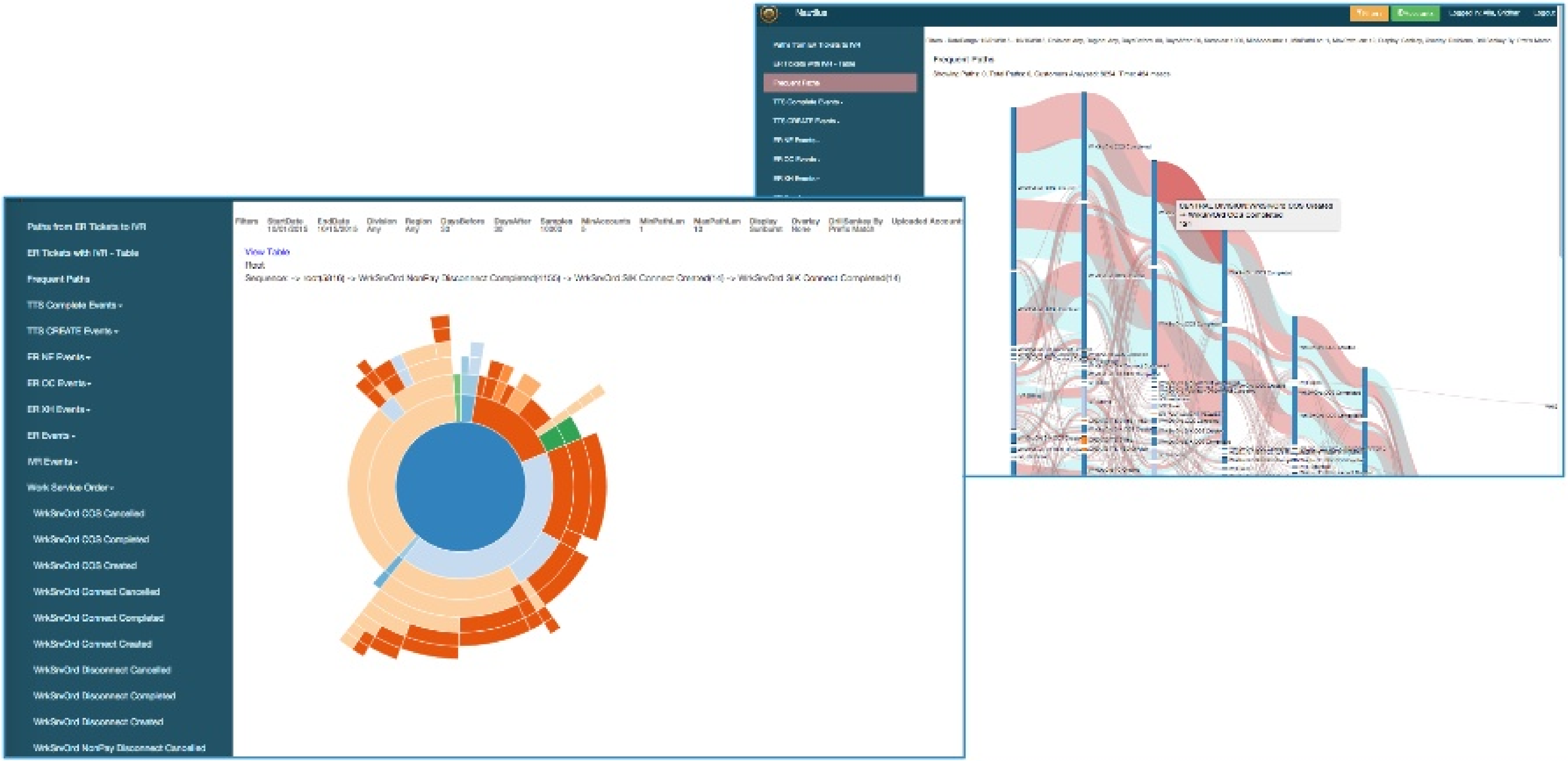
many more ………….

Direct Mail Campaign Optimization



SPARK SUMMIT 2017

Customer Journey Analytics



SPARK SUMMIT 2017

Main Challenges of Data Science

- Feature Engineering
  - Making sense of variety in data

- Model Scoring
  - Implementing ML algorithms

- Operational consumption for Business use cases

Main Challenges of Data Science

- Data ingestion, profiling and quality control
- We store and process massive amounts of data, still lack critical ability to stitch together pieces of data to make meaningful predictions. This is due to
  - Massive data size
  - Lack of service level architecture
- Multiple teams working on the same dataset
  - Increase development time because everyone has to process/feature engineer same dataset

What we needed

- A Central Processing System
  - Highly Scalable
  - Persisted and Cached
  - SQL capabilities and connection with multiple data sources and databases
  - Statistical Process Control methodology for data quality at every stage
  - Machine Learning capabilities and connection with multiple ML tools
  - Multi Tenancy
  - Access through APIs and programming languages
  - Fully automated workflow management for data science operations

What we built

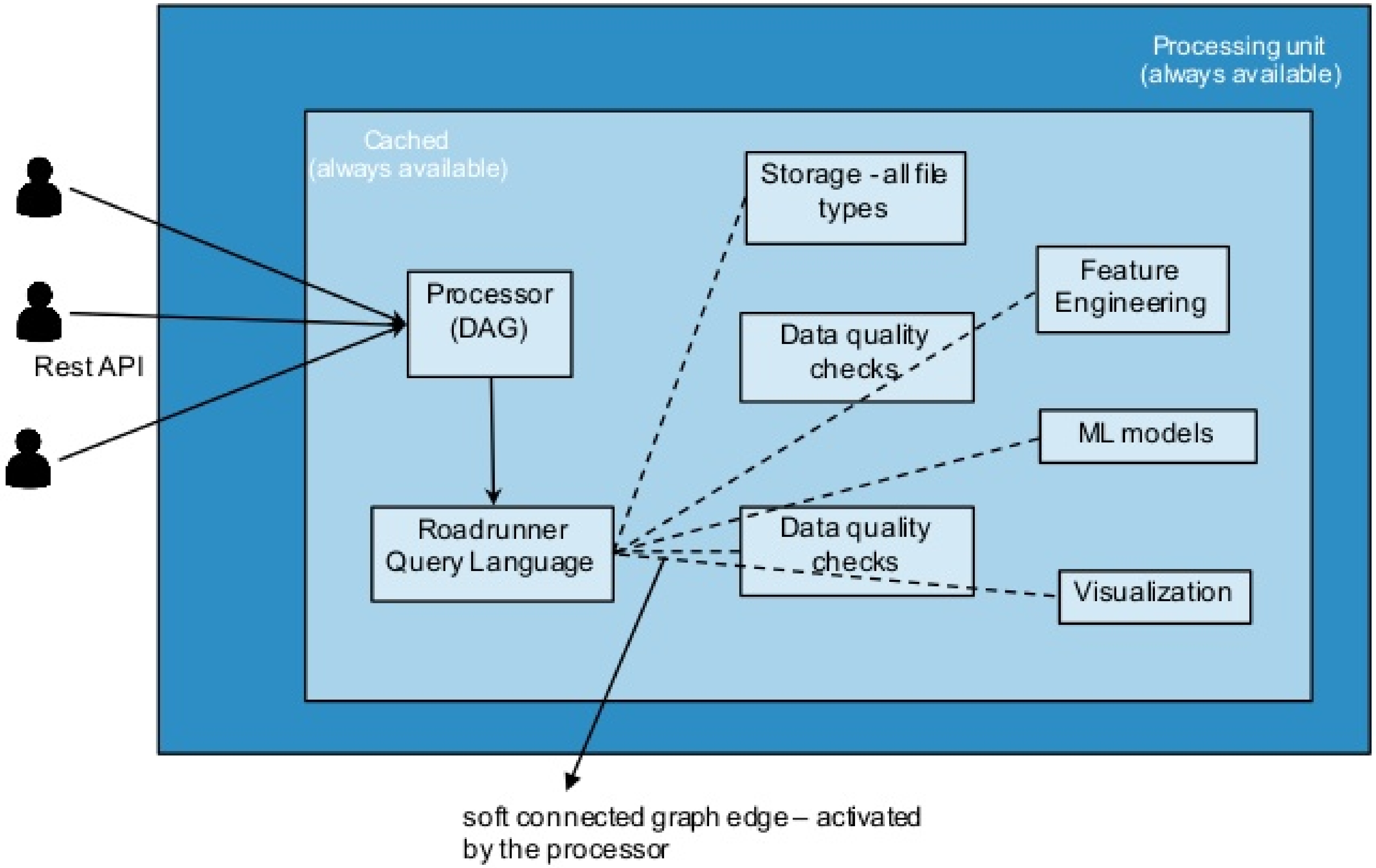
- Perpetual Spark Engine
- RESTful API to control all aspects
- Massively parallel quality control of petabyte scale datasets
    - Use Statistical Process Control methodology to check data at the record level
    - Parallelized data profilers on blind datasets
- Connectors to
    - Cassandra, Hbase, MongoDB, Teradata, MySQL, Hive, Elasticsearch, etc
    - Kafka, Storm for streaming data
    - ORC, Parquet, text files

SPARK SUMMIT 2017

What we built

- Role based control on who sees what
- Integration with modeling using Python, R, SAS, SparkML, H2O with language conversion tools
- Automated workflow management using graph methodology for data science

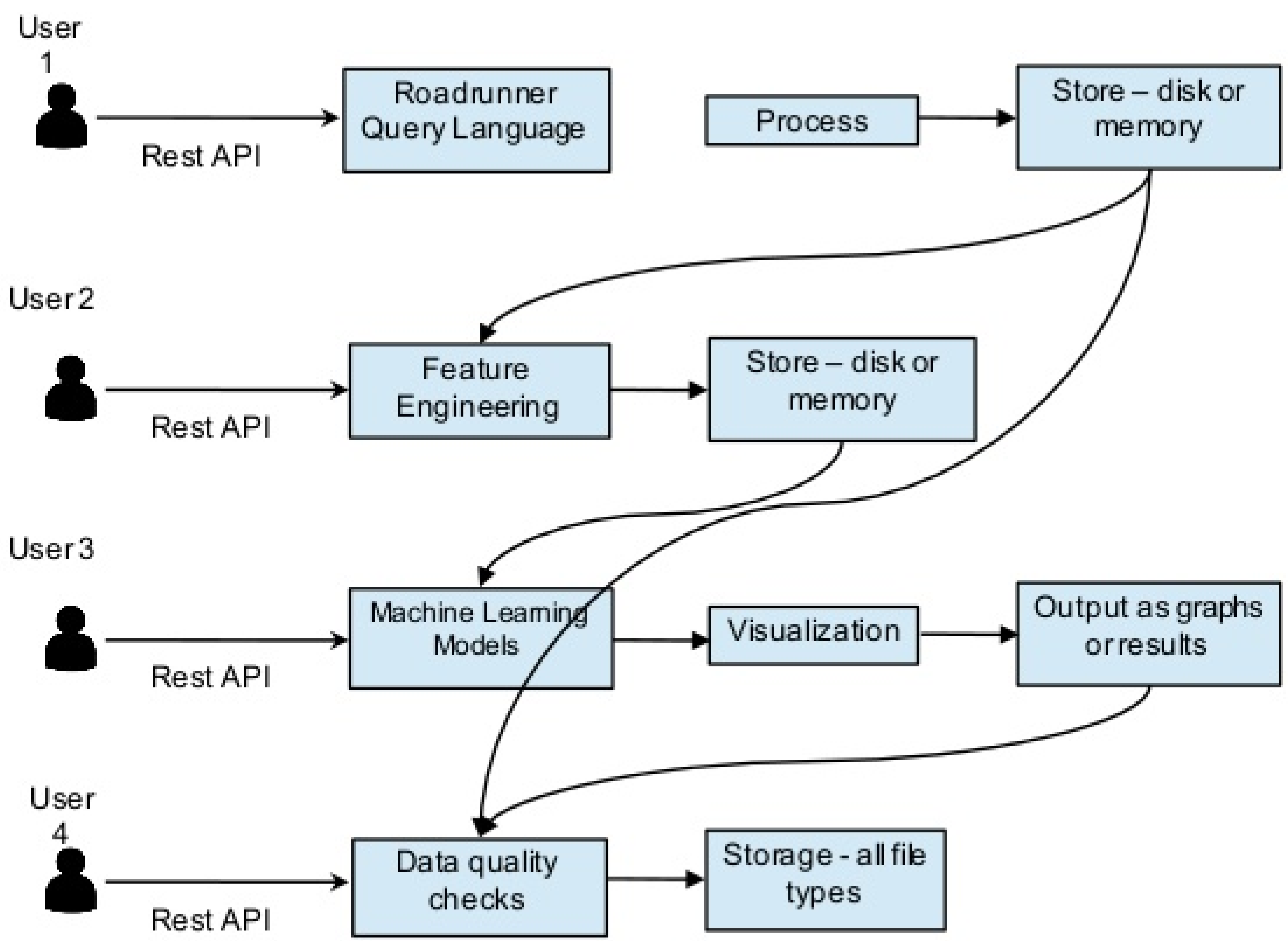
Roadrunner



Who can use Roadrunner

- Data Scientist
- DevOps
- Validation
- Modeler
- Engineer

How Roadrunner Works



SPARK SUMMIT 2017

Sample Rest API

Processing → Rest API

Feature Engineering → Rest API

Modeling → Rest API


```
{
  "jobType": "nautilusPathsJob",
  "jobId" : "JobId4",
  "rosettaTableName":"base.adm_meld_201607",
    "startTime":"2016-01-01 00:00:00",
      "endTime":"2016-02-01 00:00:00",
   "eventId":"ANY",
   "appendToEventId":"",
   "minAccounts": 1,
     "accountFilters": "ALL",
     "eventRules": {
            "condition": "OR",
            "rules" : [
              {
              "ruleType" : 2,
              "firstEventId": "ER.*",
              "secondEventId": "IVR.*",
              "op" : "gt",
              "threshold": 3,
              "timeGap" : 166400,
              "generateRuleSequences":true,
              "overlappingSequences":true,
              "exactMatchingEvents":true
              }
            ]
          }
}
```

Sample Rest API

Processing → Rest API

Feature Engineering → Rest API

Modeling → Rest API


```
{
  "jobType": "motoJob",
  "jobId" : "moto1",
  "campaignName" : "NE_Explore",
  "campaignCondition":
                    {
    "division":"NORTHEAST DIVISION",
    "channels":"tbd_mailed"
  },
  "jobStage" : "updateNode",
  "category":{"romi":">=0","connects":"2010"}
}
```


SPARK SUMMIT 2017

Examples of Transformations


```
{
  "rules": [
    {"name":"tolowercase", "colname":"regionalowercase", columns": ["SERLOC_CURRENT_REGION_NAME"]},
    {"name":"touppercase", "columns": ["SERLOC_CURRENT_DIVISION_NAME", "SERLOC_CURRENT_REGION_NAME"]},
    {"name":"concat", "columns": ["SERLOC_CURRENT_DIVISION_NAME", "SERLOC_CURRENT_REGION_NAME"]},
    {"name":"arithexpr", "expr": "(DAYSSINCE + (DAYSSINCE * CURRDAYSSINCE))", "columns": ["DAYSSINCE", "CURRDAYSSINCE"]},
    {"name":"zscore", "columns": ["DAYSSINCE"]},
    {"name":"filter", "filters": ["DAYSSINCE > 122"]},
    {"name":"decile", "colname":"DN", "scoreColumn": "DAYSSINCE", "columns": ["ACCOUNTSTATUS"]},
    {"name":"nothing", "columns": [""]}
  ]
}
```

Examples of Joins

```
{
  "rules": [
    {"name":"innerjoin", "joins": ["ACCOUNTSTATUS, astatusname"]},
    {"name":"leftouterjoin", "joins": ["ACCOUNTSTATUS, astatusname"]},
    {"name":"rightouterjoin", "joins": ["ACCOUNTSTATUS, astatusname"]},
    {"name":"outerjoin", "joins": ["ACCOUNTSTATUS, astatusname"]},
    {"name":"nothing", "joins": [""]}
  ]
}
```

Examples of Joins


```
{
  "results": [
    {
      "joinType": "inner",
      "joinTotal": 301,
      "leftTotal": 1000,
      "rightTotal": 2,
      "leftNulls": 0,
      "rightNulls": 0
    },
    {
      "joinType": "leftouter",
      "joinTotal": 1000,
      "leftTotal": 1000,
      "rightTotal": 2,
      "leftNulls": 383,
      "rightNulls": 699
    },
    {
      "joinType": "rightouter",
      "joinTotal": 302,
      "leftTotal": 1000,
      "rightTotal": 2,
      "leftNulls": 1,
      "rightNulls": 0
    },
    {
      "joinType": "outer",
      "joinTotal": 1001,
      "leftTotal": 1000,
      "rightTotal": 2,
      "leftNulls": 384,
      "rightNulls": 699
    }
  ]
}
```

Examples of Aggregations

```
{
  "rules":[ {
  "functions" : [ {
    "name" : "approx_count_distinct"
  }, {
    "name" : "histogram_string",
    "buckets" : 20
  }
  ],
  "groupBy":"SNAPSHOTDATE",
  "columns" : [ "SERLOC_CURRENT_DIVISION_NAME" ]
},
  {
    "functions" : [ {
      "name" : "approx_count_distinct"
    }, {
      "name" : "min"
    }, {
      "name" : "var_pop"
    }, {
      "name" : "sum"
    }, {
      "name" : "percentile_approx",
      "percentiles" : [ 0.25, 0.5, 0.75, 0.9, 0.95, 0.99 ]
    }
    ],
    "groupBy":"SNAPSHOTDATE,SERLOC_CURRENT_DIVISION_NAME,SERLOC_CURRENT_REGION_NAME",
    "columns" : [ "DAYSSINCE" ]
  }
]}
```

SPARK SUMMIT 2017

Examples of Aggregations


```
{
  "results": {
    "approx_count_distinct(6)": 10,
    "min(6)": "174.56",
    "max(6)": "995.09",
    "avg(6)": 566.268,
    "count(6)": 10,
    "first(6)": "706.99",
    "last(6)": "995.09",
    "kurtosis(6)": -1.4016327462769833,
    "skewness(6)": 0.12851131741001862,
    "stddev(6)": 298.53423354040245,
    "stddev_pop(6)": 283.21444125609133,
    "variance(6)": 89122.68859555555,
    "var_pop(6)": 80210.419736,
    "sum(6)": 5662.68,
    "percentile_approx(CAST(6 AS DOUBLE), array(0.25, 0.5, 0.75, 0.9, 0.95, 0.99))": [
      272.63,
      506.58,
      885.62,
      911.8,
      995.09,
      995.09
    ]
  }
}
```


SPARK SUMMIT 2017

Examples of Aggregations


```
{
  "Aggregations" : [ {
    "SNAPSHOTDATE" : "20170201",
    "SERLOC_CURRENT_DIVISION_NAME" : "NORTHEAST DIVISION",
    "SERLOC_CURRENT_REGION_NAME" : "KEYSTONE REGION",
    "approx_count_distinct(SERLOC_CURRENT_DIVISION_NAME)" : 1,
    "approx_count_distinct(DAYSSINCE)" : 33,
    "stddev(DAYSSINCE)" : 3611.7773139718893,
    "stddev_pop(DAYSSINCE)" : 3565.171783171744,
    "variance(DAYSSINCE)" : 1.3044935365721995E7,
    "var_pop(DAYSSINCE)" : 1.2710449843523994E7,
    "sum(DAYSSINCE)" : 84277,
    "percentile_approx(CAST(DAYSSINCE AS DOUBLE), array(0.25, 0.5, 0.75, 0.9, 0.95, 0.99))" : [ 112.0, 552.0, 2218.0,
  }, {
    "SNAPSHOTDATE" : "20170101",
    "SERLOC_CURRENT_DIVISION_NAME" : "NORTHEAST DIVISION",
    "SERLOC_CURRENT_REGION_NAME" : "BELTWAY REGION",
    "approx_count_distinct(SERLOC_CURRENT_DIVISION_NAME)" : 1,
    "approx_count_distinct(DAYSSINCE)" : 28,
    "stddev(DAYSSINCE)" : 1183.8048117688786,
    "stddev_pop(DAYSSINCE)" : 1170.8666657885606,
    "variance(DAYSSINCE)" : 1401393.83236715,
    "var_pop(DAYSSINCE)" : 1370928.7490548207,
    "sum(DAYSSINCE)" : 28387,
    "percentile_approx(CAST(DAYSSINCE AS DOUBLE), array(0.25, 0.5, 0.75, 0.9, 0.95, 0.99))" : [ 0.0, 60.0, 810.0, 1738
  } ],
  "customAggregations" : [ {
    "histogram_string(SERLOC_CURRENT_DIVISION_NAME)" : {
      "NORTHEAST DIVISION" : 364,
      "CENTRAL DIVISION" : 435,
      "WEST DIVISION" : 435
    }
  } ]
}
```

Deciles – Spark + Scala

```
val filters =
  dfTransformed
    .groupBy(column)
    .count
    .distinct
    .rdd
    .map(r => r.getString(0))
    .collect

val rdds = for { f <- filters } yield {
  val dfTmp = {
    if (f == null)
      dfTransformed.filter(col(column).isNull)
    else
      dfTransformed.filter(col(column) === f)
  }

  val bw = Window.partitionBy(column).orderBy(col(scoreColumn).desc)
  val df2 = dfTmp.select(col("*"), ntile(10).over(bw).alias(colName.getOrElse("decile")))
```

Deciles – the Roadrunner way…

```
{
  "rules": [
    {"name":"decile", "colname":"DN", "scoreColumn": "DAYSSINCE", "columns": ["ACCOUNTSTATUS"]}
  ]
}
```

SPARK SUMMIT 2017

Grouped Aggregations – easy?

```scala
val (aggColumnFunctions, toCalculate) = columnFunctions.partition(
  _.function.isDefined
) //if a columnFunction has a function=None then it is a custom function and cannot be handled by the `df.agg` call
val aggFunctions = aggColumnFunctions.flatMap(_.function)

def columnNames(funcs: Seq[ColumnFunction]) =
  funcs.flatMap(_.columnNames).distinct.map(col)

def groupByColumnsFunc(funcs: Seq[ColumnFunction]) =
  funcs.flatMap(_.groupBy.getOrElse("").split(",").map(x => x.trim)).distinct.map(col)

val groupByColumns = groupByColumnsFunc(aggColumnFunctions)

val resultsF = Future(blocking {

  aggFunctions.isEmpty.fold(
    { List() }, {
      logger.debug(s"Running df.agg(${aggFunctions.mkString(",")}")
      if (groupByColumns.isEmpty) {
        df.select(columnNames(aggColumnFunctions): _*)
          .agg(aggFunctions.head, aggFunctions.tail: _*)
          .toJSON
          .collectAsList()
          .toList
      } else {
        df.groupBy(groupByColumnsFunc(aggColumnFunctions): _*)
          .agg(aggFunctions.head, aggFunctions.tail: _*)
          .toJSON
          .collectAsList()
          .toList
      }
    }
  )
})
```

SPARK SUMMIT 2017

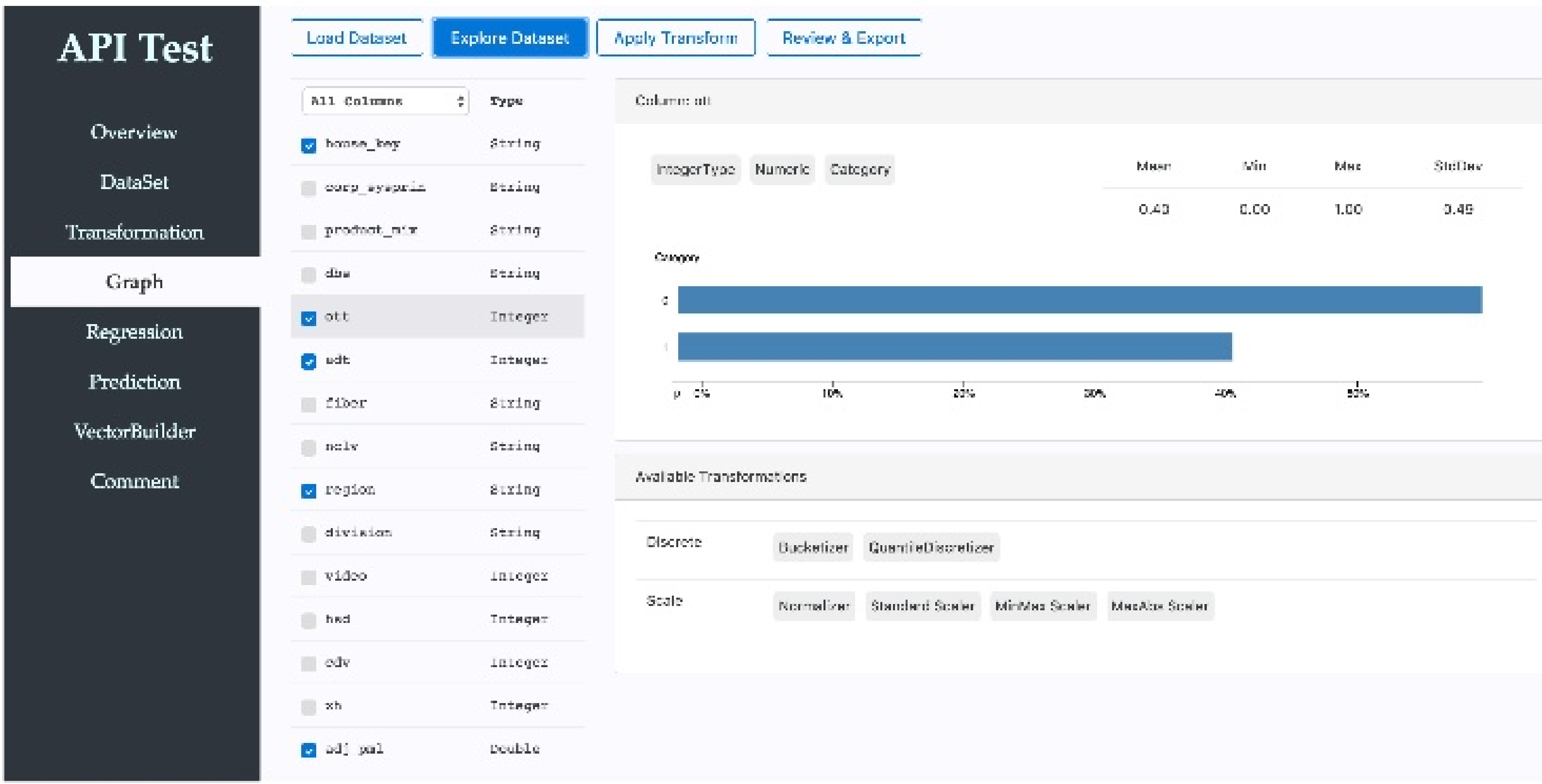
Grouped Aggregations – easy?

```
val (aggColumnFunctions, toCalculate) = columnFunctions.partition(
  _.function.isDefined
) //if a columnFunction has a function=None then it is a custom function and cannot be handled by the `df.agg` call
val aggFunctions = aggColumnFunctions.flatMap(_.function)

def columnNames(funcs: Seq[ColumnFunction]) =
  funcs.flatMap(_.columnNames).distinct.map(col)

def groupByColumnsFunc(funcs: Seq[ColumnFunction]) =
  funcs.flatMap(_.groupBy.getOrElse("").split(",").map(x => x.trim)).distinct.map(col)

val groupByColumns = groupByColumnsFunc(aggColumnFunctions)

val resultsF = Future(blocking {

  aggFunctions.isEmpty.fold(
    { List() }, {
      logger.debug(s"Running df.agg(${aggFunctions.mkString(",")}")
      if (groupByColumns.isEmpty) {
        df.select(columnNames(aggColumnFunctions): _*)
          .agg(aggFunctions.head, aggFunctions.tail: _*)
          .toJSON
          .collectAsList()
          .toList
      } else {
        df.groupBy(groupByColumnsFunc(aggColumnFunctions): _*)
          .agg(aggFunctions.head, aggFunctions.tail: _*)
          .toJSON
          .collectAsList()
          .toList
      }
    }
  )
})
```

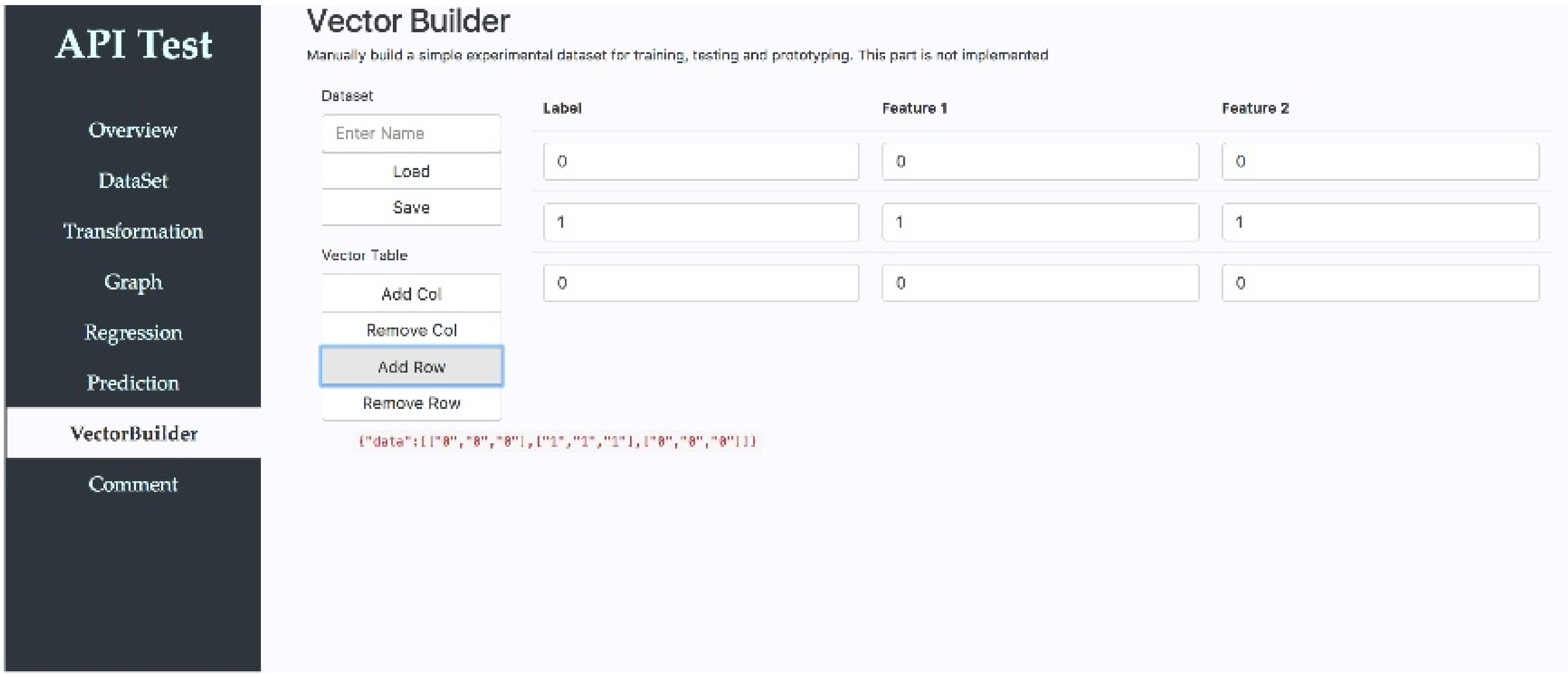
Grouped Aggregations - the Roadrunner way

```
{
  "rules":[ {
  "functions" : [ {
    "name" : "approx_count_distinct"
  }, {
    "name" : "histogram_string",
    "buckets" : 20
  }
  ],
  "groupBy":"SNAPSHOTDATE",
  "columns" : [ "SERLOC_CURRENT_DIVISION_NAME" ]
},
  {
    "functions" : [ {
      "name" : "approx_count_distinct"
    }, {
      "name" : "min"
    }, {
      "name" : "max"
    }, {
      "name" : "avg"
    }, {
      "name" : "stddev"
    }, {
      "name" : "stddev_pop"
    }, {
      "name" : "var_pop"
    }, {
      "name" : "sum"
    }, {
      "name" : "percentile_approx",
      "percentiles" : [ 0.25, 0.5, 0.75, 0.9, 0.95, 0.99 ]
    }
    ],
    "groupBy":"SNAPSHOTDATE,SERLOC_CURRENT_DIVISION_NAME,SERLOC_CURRENT_REGION_NAME",
    "columns" : [ "DAYSSINCE" ]
  }
]}
```

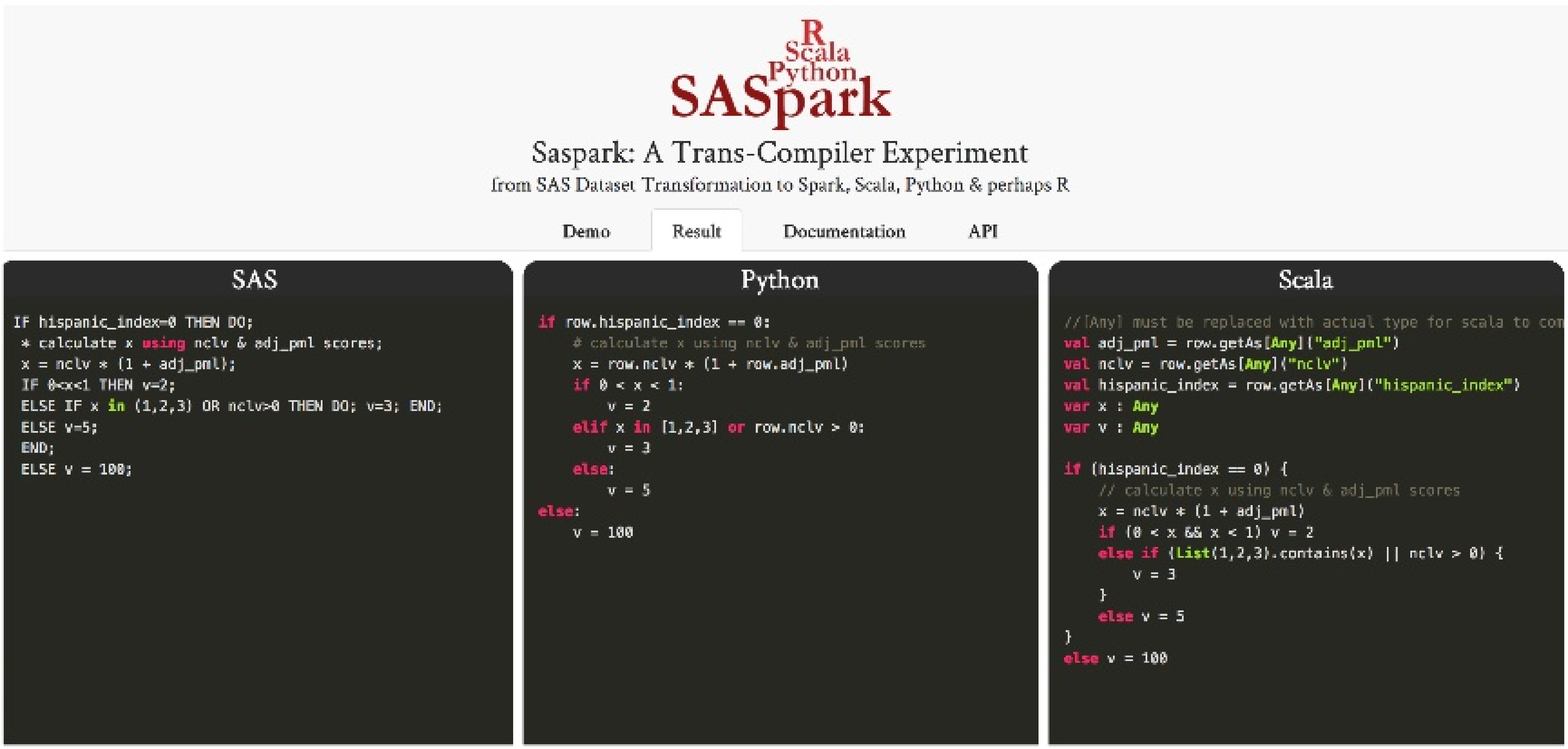
Exploration



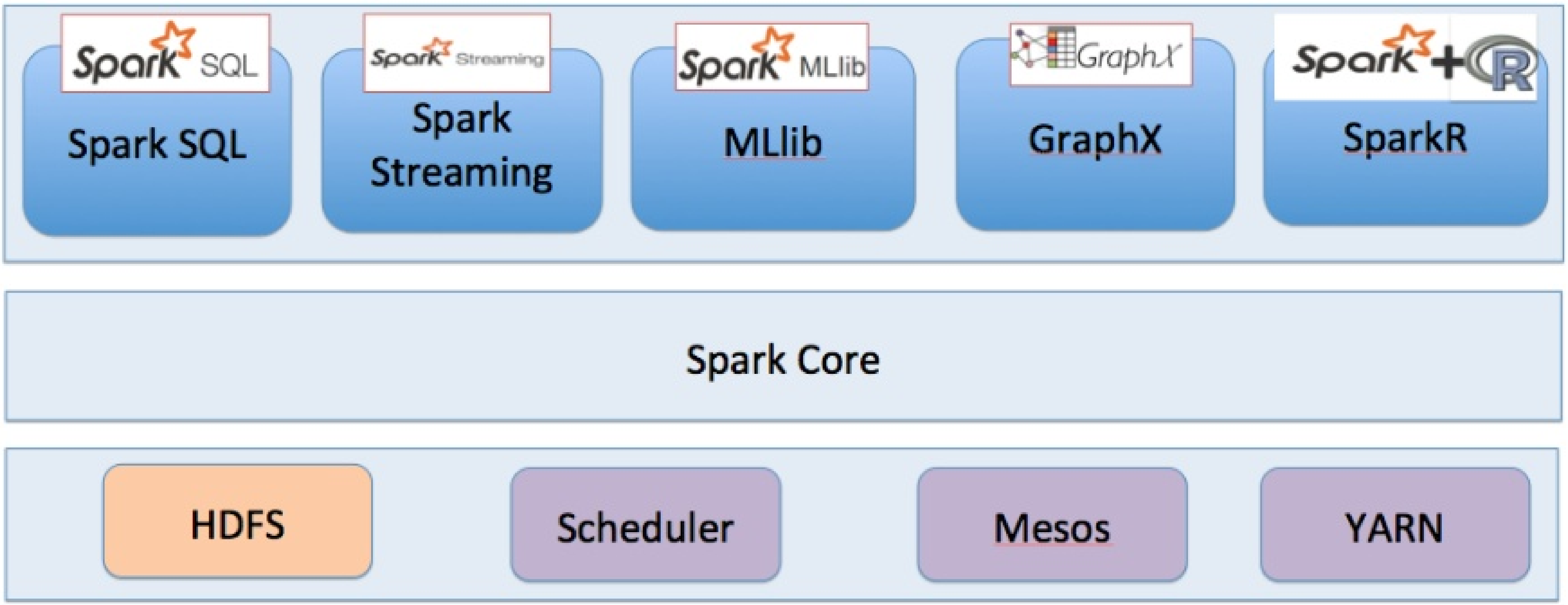
Features


API Test
Overview
DataSet
Transformation
Graph
Regression
Prediction
VectorBuilder
Comment
Vector Builder
Manually build a simple experimental dataset for training, testing and prototyping. This part is not implemented
Dataset
Enter Name
Load
Save
Vector Table
Add Col
Remove Col
Add Row
Remove Row
Label
Feature 1
Feature 2
0
0
0
1
1
1
0
0
0
{"data":[["0","0","0"],["1","1","1"],["0","0","0"]]}

Converter


R
Scala
Python
SASpark
Saspark: A Trans-Compiler Experiment
from SAS Dataset Transformation to Spark, Scala, Python & perhaps R
Demo
Result
Documentation
API
SAS
IF hispanic_index=0 THEN DO;
* calculate x using nclv & adj_pml scores;
x = nclv * (1 + adj_pml);
IF 0<x<1 THEN v=2;
ELSE IF x in (1,2,3) OR nclv>0 THEN DO; v=3; END;
ELSE v=5;
END;
ELSE v = 100;
Python
if row.hispanic_index == 0:
# calculate x using nclv & adj_pml scores
x = row.nclv * (1 + row.adj_pml)
if 0 < x < 1:
v = 2
elif x in [1,2,3] or row.nclv > 0:
v = 3
else:
v = 5
else:
v = 100
Scala
//[Any] must be replaced with actual type for scala to com
val adj_pml = row.getAs[Any]("adj_pml")
val nclv = row.getAs[Any]("nclv")
val hispanic_index = row.getAs[Any]("hispanic_index")
var x : Any
var v : Any
if (hispanic_index == 0) {
// calculate x using nclv & adj_pml scores
x = nclv * (1 + adj_pml)
if (0 < x && x < 1) v = 2
else if (List(1,2,3).contains(x) || nclv > 0) {
v = 3
}
else v = 5
}
else v = 100

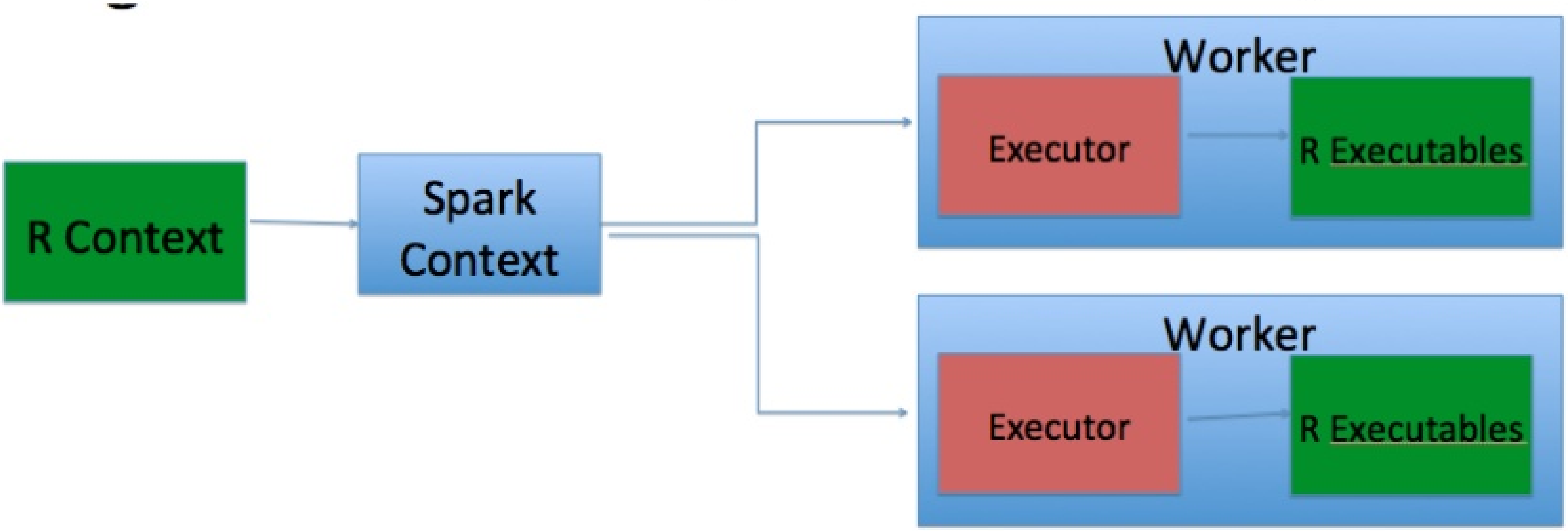
Spark Stack



SPARK SUMMIT 2017

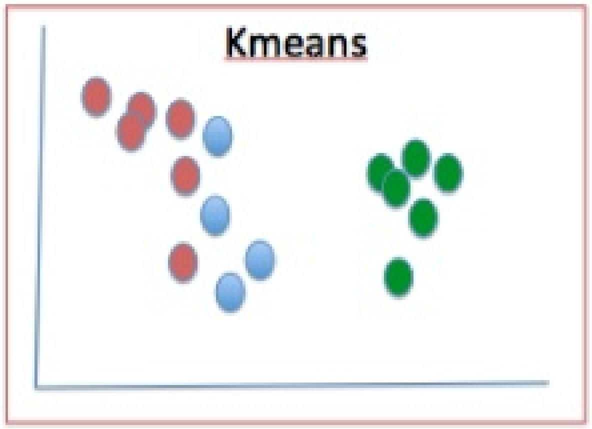
SparkR

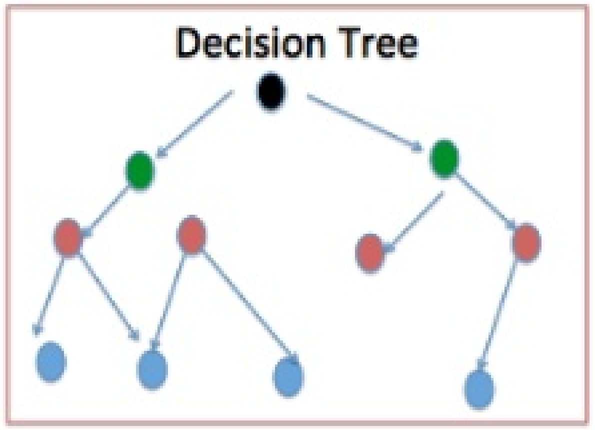
- Enables using R packages to process data
- Can run Machine Learning and Statistical Analysis



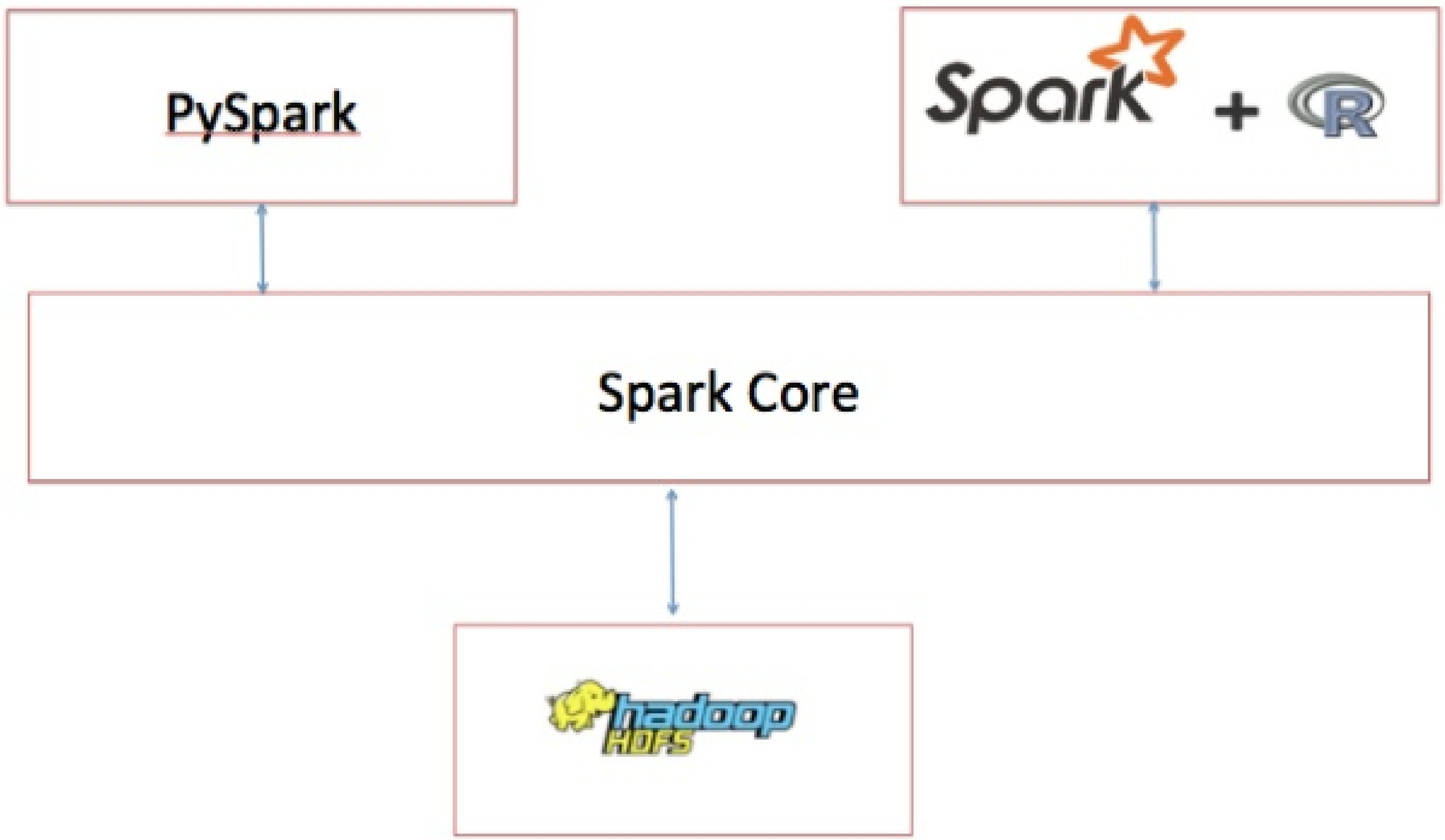
Spark MLlib

- Implements various Machine Learning Algorithms
- Classification, Regression, Collaborative Filtering, Clustering, Decomposition
- Works with Streaming, Spark SQL, GraphX or with SparkR.





PySpark, TensorFlow and SparkR



SPARK SUMMIT 2017

We are hiring!

- Big Data Engineers (Hadoop, Spark, Kafka…)
- Data Analysts (R, SAS…..)
- Big Data Analysts (Hive, Pig ….)

jobs.comcast.com

Thank You.

SPARK SUMMIT 2017

Data Science Initiatives

- Customer Churn Prediction
- Click-thru Analytics
- Personalization
- Customer Journey
- Modeling
- Anomaly Detection
- GPU driven optimizations

Anomaly Detection

- Identification of observations which do not conform to an expected pattern.
- Ex: Network Intrusion Detection, Spikes in operational data, Unusual usage activity.

Popular Algorithms

- Unsupervised
  - KMeans
  - DBScan
- Supervised
  - HMM
  - Neural networks

KMeans Clustering

- Clustering is an unsupervised learning problem
- Groups subsets of entities with one another based on some notion of similarity.
- Easy to check if a new entity is falling outside known groups/clusters

Sample Code

```
import org.apache.spark.mllib.clustering.KMeans
import org.apache.spark.mllib.linalg.Vectors

val lines = sc.textFile("training.csv")
val data = lines.map(line => line.split(",").map(_.trim))
val inData = data.map{(x) => (x(3)) }.map(_.toLong)
val inVector = inData.map{a => Vectors.dense(a)}.cache()
val numClusters = 3
val numIterations = 100
val kMeans = new KMeans().setMaxIterations(numIterations).setK(numClusters)
val kMeansModel = kMeans.run(inVector)

// Print cluster index for a given observation point
var ci = kMeansModel.predict(Vectors.dense(10000.0))
var ci = kMeansModel.predict(Vectors.dense(900008830.0))
```

Sample Code (R):

```
library('RHmm')
indata <- read.csv(file.choose(), header = FALSE, sep = ",", quote = "\"", dec = ".")
testdata <- read.csv(file.choose(), header = FALSE, sep = ",", quote = "\"", dec = ".")
dataSets <- c(as.numeric(indata$V4))
dataSetModel <- HMMFit(dataSets, nStates=3)
testdataSets <- c(as.numeric(testdata$V4))
tVitPath <- viterbi(dataSetModel, testdataSets)

#Forward-backward procedure, compute probabilities
tfb <- forwardBackward(dataSetModel, testdataSets)

# Plot implied states
layout(1:3) dataSet
plot(testdataSets[1:100],ylab="StateA",type="l", main="dataSet A")
plot(tVitPath$states[1:100],ylab="StateB",type="l", main="dataSet B")
```

Add Slides as Necessary

- Supporting points go here.

SPARK
SUMMIT
2017

Thank You.

Contact information or call to action goes here.