SPARK+AI SUMMIT 2019

WIFI SSID: SparkAISummit | Password: UnifiedAnalytics

ORGANIZED BY databricks

Apache Spark Data Validation

Patrick Pisciuneri & Doug Balog, Target

#UnifiedAnalytics #SparkAISummit

Outline

- Introductions
- Motivation
- Data Validator
- Open Source
- Future Work
- Conclusions

Introduction

Patrick Pisciuneri

Patrick is a Data Scientist for Target. He received his PhD in Mechanical Engineering from the University of Pittsburgh in 2013. His research involved the intersection of high-performance computing and the simulation of turbulent reacting flows. In 2015 he joined Target as a data scientist where he has worked on product and ad recommendations.

Doug Balog

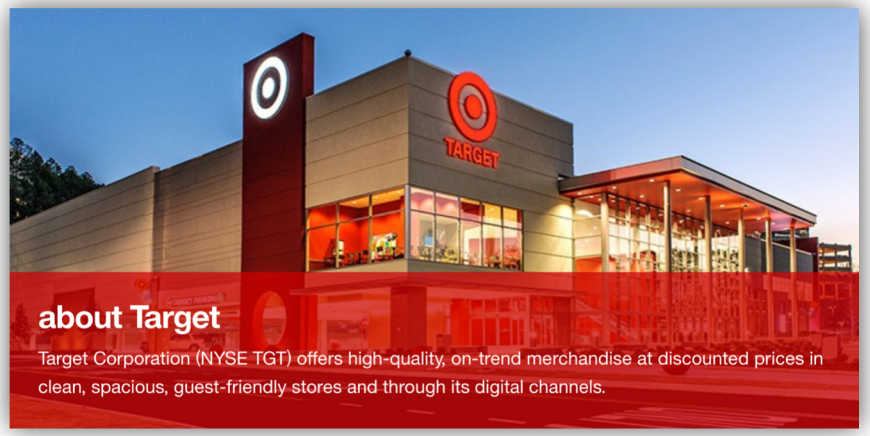
Doug is a Data Engineer for Target. He develops Machine Learning infrastructure for Target in Pittsburgh, PA. He joined Target in 2014 and is currently a Principal Data Engineer. He has a BS in Computer Science from University of Pittsburgh.

Introduction

Target

- 1,845 stores in the United States
- 39 distribution centers in the United States
- 350,000+ team members worldwide
- Online business at target.com
- Global offices in China, Hong Kong and India

corporate.target.com/



Motivation

- Understand data
- Catch errors/anomalies early in pipelines
- Promote best practices for data management
- Easy to adopt – language agnostic interface
- Efficient on large datasets and distributed systems

Motivation

References

1. *TFX: A TensorFlow-Based Production-Scale Machine Learning Platform*, KDD 2017
2. *Hidden Technical Debt in Machine Learning*, NIPS 2015
3. *Methodology for Data Validation 1.0*, Eurostat – CROS, 2016
4. *Extending Apache Spark APIs Without Going Near Spark Source or a Compiler*, Spark + AI Summit, 2018

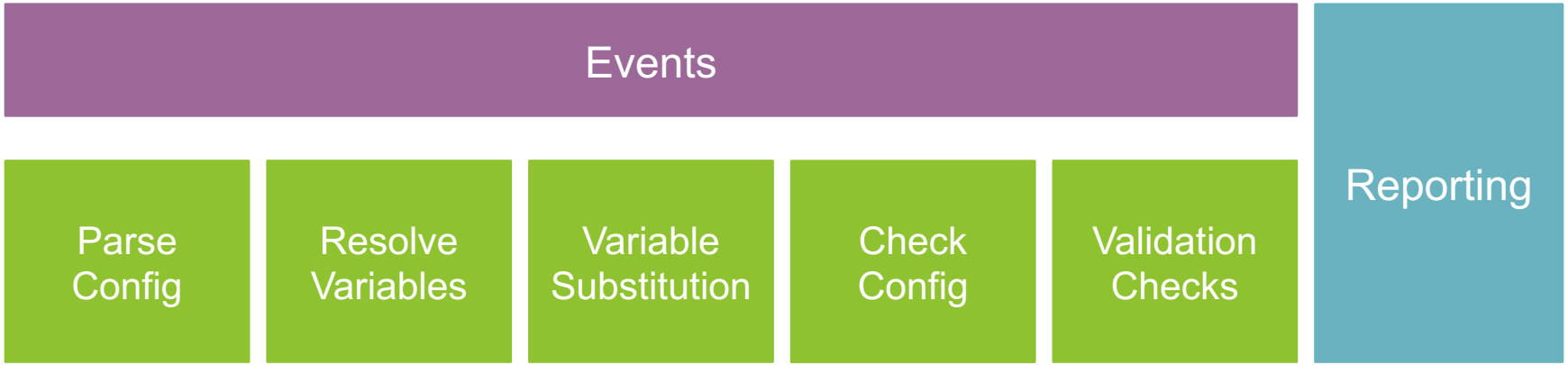
introducing

Data Validator

Features

- Configuration
- Validators
- Reporting & Notification
- Profiling

Flow Diagram



Configuration

- User creates a configuration file (yaml)
  1. Variables
  2. Reporting settings
  3. Notification settings
  4. Validation checks

Configuration - Variables

- Set the value of a variable specified by the name value
  1. Simple substitution
  2. Environment variable substitution
  3. Shell substitution
     - First line of stdout
  4. SQL substitution
     - First column of first row

```
vars:
  - name: ENV
    value: prod

  - name: JAVA_DIR
    env: JAVA_HOME

  - name: TODAY
    shell: date + "%Y-%m-%d"

  - name: MIN_AGE
    sql: SELECT min(age) FROM adult
```

```
outputs:
  - filename: /user/home/sample.json
    append: true
```

Configuration – Report Settings

- Event report
  - Specify path and filename
    - local://
    - hdfs://
    - **append** – will overwrite existing file if false
  - Pipe event report to another program
    - **ignoreError** – mark data validator as failed if pipe command fails

```
    shell: date + "%Y-%m-%d"

  - name: MIN_AGE
    sql: SELECT min(age) FROM adult

outputs:
  - filename: /user/home/sample.json
    append: true

  - pipe: /path/to/program
    ignoreError: true

email:
  smtpHost: smtp.example.com
  subject: Data Validation Summary
  from: data-validator-no-reply@exmple.com
  to:
```

Configuration – Notification Settings

- Send email with status of validation checks

Validator Report

HiveTable:`census_income.adult` with condition(educationNum >= 5)

FAIL - MinNumRowCheck 50000 count:31363 errors:1 pct:0.00%
PASS - NegativeCheck(age)
PASS - NullCheck(occupation)

```
  - pipe: /path/to/program
    ignoreError: true

email:
  smtpHost: smtp.example.com
  subject: Data Validation Summary
  from: data-validator-no-reply@exmple.com
  to:
    - person1@example.com
  cc:
    - person2@example.com, person3@example.com
  bcc:
    - person4@example.com

tables:
  - db: census_income
    table: adult
```

Configuration – Validation Checks

- Specify the checks we want to run on a data asset
- Support:
  - Hive tables
  - Orc files
  - Parquet files
- Any number of checks on any number of columns
- Specify **`condition`** to filter asset
  - Useful when applied to partition column

```
    - person2@example.com, person3@example.com
  bcc:
    - person4@example.com

tables:
  - db: census_income
    table: adult
    condition: educationNum >= 5
    checks:
      - type: rowCount
        minNumRows: 50000

      - type: negativeCheck
        column: age

      - type: nullCheck
        column: occupation
```

ValidatorBase

```
abstract class ValidatorBase(
  var failed: Boolean = false,
  events: ListBuffer[ValidatorEvent] = new ListBuffer[ValidatorEvent]
) extends Substitutable {
  import ValidatorBase._

  def name: String = this.getClass.getSimpleName

  def substituteVariables(dict: VarSubstitution): ValidatorBase

  def configCheck(df: DataFrame): Boolean

  def select(schema: StructType, dict: VarSubstitution): Expression

  def quickCheck(r: Row, count: Long, idx: Int): Boolean

  def generateHTMLReport: Tag = {...}

  def toJson: Json

  // Utility methods

  final def addEvent(ve: ValidatorEvent): Unit = {
    events.append(ve)
    failed ||= ve.failed
  }
```

RowBased

```
abstract class RowBased extends ValidatorBase {

  val column: String

  def configCheck(df: DataFrame): Boolean = configCheckColumn(df)

  def configCheckColumn(df: DataFrame): Boolean = {...}

  def colTest(schema: StructType, dict: VarSubstitution): Expression

  def select(schema: StructType, dict: VarSubstitution): Expression = If(colTest(schema, dict), L1, L0)

  override def quickCheck(row: Row, count: Long, idx: Int): Boolean = {...}

  def quickCheckDetail(row: Row, key: Seq[(String, Any)], idx: Int, dict: VarSubstitution): Unit = {...}
}

case class NullCheck(column: String) extends RowBased {...}

case class NegativeCheck(column: String) extends RowBased {...}

case class RangeCheck(
  column: String,
  minValue: Option[Json],
  maxValue: Option[Json],
  inclusive: Option[Json]
) extends RowBased {...}
```

ColumnBased

```
abstract class ColumnBased(column: String, condTest: Expression) extends ValidatorBase {
  override def select(schema: StructType, dict: VarSubstitution): Expression = condTest

  // ColumnBased checks don't have per row error details.
  def hasQuickErrorDetails: Boolean = false
}

case class MinNumRows(minNumRows: Long) extends ColumnBased("", ValidatorBase.L0) {...}

case class ColumnMaxCheck(column: String, value: Json)
  extends ColumnBased(column, Max(UnresolvedAttribute(column)).toAggregateExpression()) {...}
```

Example

- Census (1994) Income Data Set
- Learning task: predict whether income exceeds 50k

http://archive.ics.uci.edu/ml/datasets/Census+Income

```
39, State-gov, 77516, Bachelors, 13, Never-married, Adm-clerical, Not-in-family, White, Male, 2174, 0, 40, United-States, <=50K
50, Self-emp-not-inc, 83311, Bachelors, 13, Married-civ-spouse, Exec-managerial, Husband, White, Male, 0, 0, 13, United-States, <=50K
38, Private, 215646, HS-grad, 9, Divorced, Handlers-cleaners, Not-in-family, White, Male, 0, 0, 40, United-States, <=50K
53, Private, 234721, 11th, 7, Married-civ-spouse, Handlers-cleaners, Husband, Black, Male, 0, 0, 40, United-States, <=50K
28, Private, 338409, Bachelors, 13, Married-civ-spouse, Prof-specialty, Wife, Black, Female, 0, 0, 40, Cuba, <=50K
37, Private, 284582, Masters, 14, Married-civ-spouse, Exec-managerial, Wife, White, Female, 0, 0, 40, United-States, <=50K
49, Private, 160187, 9th, 5, Married-spouse-absent, Other-service, Not-in-family, Black, Female, 0, 0, 16, Jamaica, <=50K
52, Self-emp-not-inc, 209642, HS-grad, 9, Married-civ-spouse, Exec-managerial, Husband, White, Male, 0, 0, 45, United-States, >50K
31, Private, 45781, Masters, 14, Never-married, Prof-specialty, Not-in-family, White, Female, 14084, 0, 50, United-States, >50K
42, Private, 159449, Bachelors, 13, Married-civ-spouse, Exec-managerial, Husband, White, Male, 5178, 0, 40, United-States, >50K
37, Private, 280464, Some-college, 10, Married-civ-spouse, Exec-managerial, Husband, Black, Male, 0, 0, 80, United-States, >50K
30, State-gov, 141297, Bachelors, 13, Married-civ-spouse, Prof-specialty, Husband, Asian-Pac-Islander, Male, 0, 0, 40, India, >50K
23, Private, 122272, Bachelors, 13, Never-married, Adm-clerical, Own-child, White, Female, 0, 0, 30, United-States, <=50K
32, Private, 205019, Assoc-acdm, 12, Never-married, Sales, Not-in-family, Black, Male, 0, 0, 50, United-States, <=50K
40, Private, 121772, Assoc-voc, 11, Married-civ-spouse, Craft-repair, Husband, Asian-Pac-Islander, Male, 0, 0, 40, ?, >50K
34, Private, 245487, 7th-8th, 4, Married-civ-spouse, Transport-moving, Husband, Amer-Indian-Eskimo, Male, 0, 0, 45, Mexico, <=50K
```

Example

Config

```
tables:
  - db: census_income
    table: adult
    condition: educationNum >= 5
    checks:
      - type: rowCount
        minNumRows: 50000

      - type: negativeCheck
        column: age

      - type: nullCheck
        column: occupation
```

Schema

```
scala> adult.printSchema
root
 |-- age: integer (nullable = true)
 |-- workclass: string (nullable = true)
 |-- finalWeight: integer (nullable = true)
 |-- education: string (nullable = true)
 |-- educationNum: integer (nullable = true)
 |-- maritalStatus: string (nullable = true)
 |-- occupation: string (nullable = true)
 |-- relationship: string (nullable = true)
 |-- race: string (nullable = true)
 |-- sex: string (nullable = true)
 |-- capitalGain: integer (nullable = true)
 |-- capitalLoss: integer (nullable = true)
 |-- hoursPerWeek: integer (nullable = true)
 |-- country: string (nullable = true)
 |-- target: string (nullable = true)
```

Explain

Config

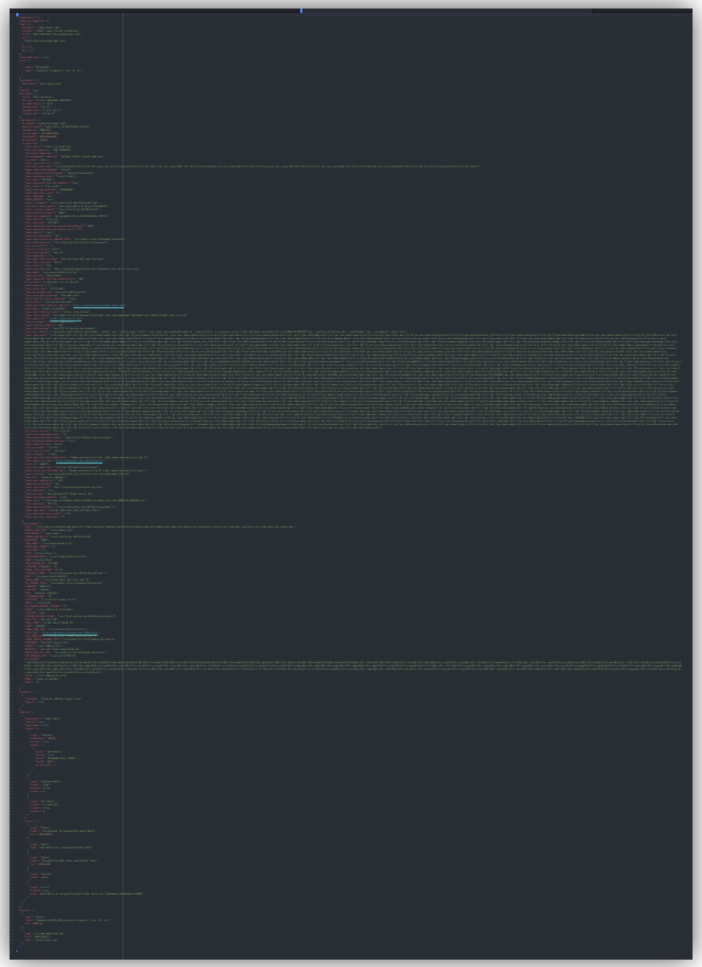
```
tables:
  - db: census_income
    table: adult
    condition: educationNum >= 5
    checks:
      - type: rowCount
        minNumRows: 50000

      - type: negativeCheck
        column: age

      - type: nullCheck
        column: occupation
```

```
== Physical Plan ==
*(2) HashAggregate(
  keys=[],
  functions=[
    count(1),
    sum(if ((age#0 < 0)) 1 else 0),
    sum(if (isnull(occupation#6)) 1 else 0)
  ]
)
+- Exchange SinglePartition
   +- *(1) HashAggregate(
     keys=[],
     functions=[
       partial_count(1),
       partial_sum(if ((age#0 < 0)) 1 else 0),
       partial_sum(if (isnull(occupation#6)) 1 else 0)
    ]
  )
      +- *(1) Project [age#0, occupation#6]
         +- *(1) Filter (
           isnotnull(educationNum#4) && (educationNum#4 >= 5)
         )
            +- *(1) FileScan parquet census_income.adult ...
```

** formatted in text editor for clarity*

Reporting

- Event log
  - Logs configuration settings
  - Logs variable substitutions
  - Logs runtime environment and statistics
  - Logs checks run and status
  - It’s extensive!



Report

Config

```
tables:
  - db: census_income
    table: adult
    condition: educationNum >= 5
    checks:
      - type: rowCount
        minNumRows: 50000

      - type: negativeCheck
        column: age

      - type: nullCheck
        column: occupation
```

rowCount

```
"keyColumns": null,
"checks": [
  {
    "type": "rowCount",
    "minNumRows": 50000,
    "failed": true,
    "events": [
      {
        "type": "checkEvent",
        "failed": true,
        "label": "MinNumRowCheck 50000 ",
        "count": 31363,
        "errorCount": 1
      }
    ]
  },
  {
```

Report

Config

```
tables:
  - db: census_income
    table: adult
    condition: educationNum >= 5
    checks:
      - type: rowCount
        minNumRows: 50000

      - type: negativeCheck
        column: age

      - type: nullCheck
        column: occupation
```

negativeCheck

```
},
{
  "type": "negativeCheck",
  "column": "age",
  "failed": false,
  "events": [
    {
      "type": "checkEvent",
      "failed": false,
      "label": "NegativeCheck on column 'age'",
      "count": 31363,
      "errorCount": 0
    }
  ]
},
{
  "type": "nullCheck",
```

Report

Config

```
tables:
  - db: census_income
    table: adult
    condition: educationNum >= 5
    checks:
      - type: rowCount
        minNumRows: 50000

      - type: negativeCheck
        column: age

      - type: nullCheck
        column: occupation
```

nullCheck

```
},
{
  "type": "nullCheck",
  "column": "occupation",
  "failed": false,
  "events": [
    {
      "type": "checkEvent",
      "failed": false,
      "label": "NullCheck on column 'occupation'",
      "count": 31363,
      "errorCount": 0
    }
  ]
}

vents": [
```

Profiling

Config

```
tables:
  - db: census_income
    table: adult
    checks:
      - type: colstats
        column: age
      - type: colstats
        column: educationNum
```

```
{
  "type": "colstats",
  "column": "age",
  "failed": false,
  "events": [
    {
      "name": "FirstPassStats",
      "column": "age",
      "count": 32561,
      "min": 17,
      "mean": 38.58164675532078,
      "max": 90
    }
  ]
},
```

Profiling

Config

```
tables:
  - db: census_income
    table: adult
    checks:
      - type: colstats
        column: age

      - type: colstats
        column: educationNum
```

```
{
  "type": "colstats",
  "column": "educationNum",
  "failed": false,
  "events": [
    {
      "name": "FirstPassStats",
      "column": "educationNum",
      "count": 32561,
      "min": 1,
      "mean": 10.0806793403151,
      "max": 16
    }
  ]
},
```

Profiling Implementation

- Catalyst aggregate expressions
  - Min
  - Max
  - Average
  - Count
  - Stddev
- UDAF
  - histogram

```
class FirstPassStatsAggregator extends UserDefinedAggregateFunction {

  override def inputSchema: StructType = new StructType().add("value", DoubleType)

  override def bufferSchema: StructType = ...

  override def dataType: DataType = FirstPassStats.dataType

  override def deterministic: Boolean = true

  override def initialize(buffer: MutableAggregationBuffer): Unit = {...}

  override def update(buffer: MutableAggregationBuffer, input: Row): Unit = {...}

  override def merge(buffer1: MutableAggregationBuffer, buffer2: Row): Unit = {...}

  override def evaluate(buffer: Row): Any = {...}

  private val count = bufferSchema.fieldIndex("count")
  private val sum = bufferSchema.fieldIndex("sum")
  private val min = bufferSchema.fieldIndex("min")
  private val max = bufferSchema.fieldIndex("max")
}
```

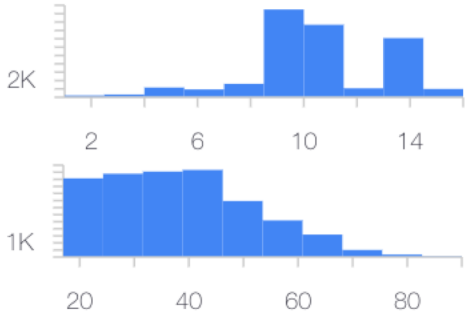
Profiling

- Parse / pipe output from report to tool for visualization
- Facets Overview

Numeric Features (2)

Chart to show: Standard ☐log ☐expand

	count	missing	mean	std dev	zeros	min	median	max
Education-Num	32.6k	0%	10.08	2.57	0%	1	10	16
Age	32.6k	0%	38.58	13.64	0%	17	37	90



Open Source

- Apache License, Version 2.0
- https://github.com/target/data-validator

Future Work

- Add more validators
- Leverage column statistics for default configuration generation
- HDP 3 and Spark 2.4+
- Anomaly detection
- Distribution shift
- Ideas and input from community

Conclusions

- Adoption
- Understand data
- Catch errors/anomalies early in pipelines
- Promote best practices for data management
- Easy to adopt – language agnostic interface
- Efficient on large datasets and distributed systems

Thank you!

- Questions?

- We are hiring:
  - jobs.target.com
  - Search “lead data scientist”, “lead data engineer”
- Check out: *Parallelizing with Apache Spark in Unexpected Ways* @ 2:40 PM by Anna Holschuh

SPARK+AI SUMMIT 2019

DON'T FORGET TO RATE AND REVIEW THE SESSIONS

SEARCH SPARK + AI SUMMIT

Download on the App Store

GET IT ON Google Play