Getting Started with Spark 2.x Machine Learning Pipelines to Predict Flight Delays

MAPR®

CAROL MCDONALD

Solutions Architect

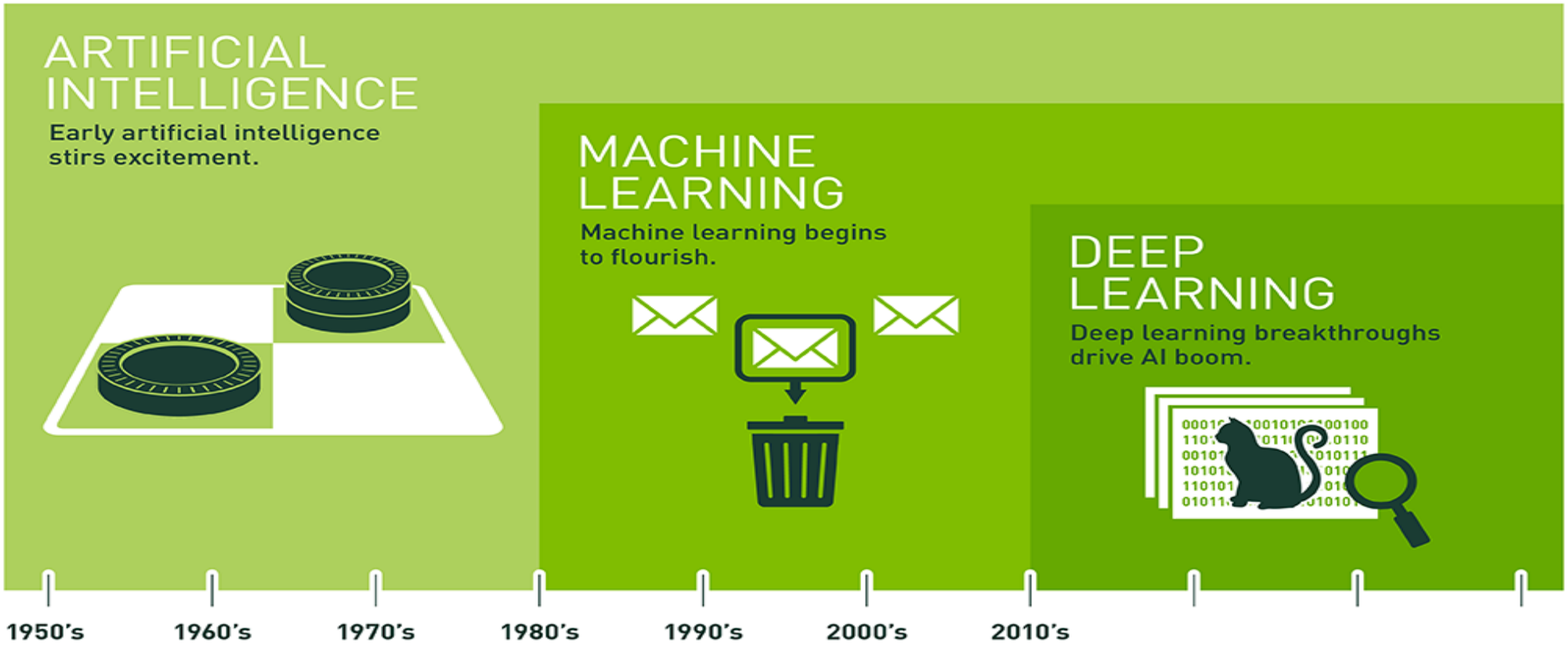
Agenda

- Introduction to Machine Learning Classification
- Explore the Flight Dataset with Spark DataFrames
- Use Random Forest to Predict Flight Delays

Intro to Machine Learning

What is AI?

The definition of AI is continuously redefined

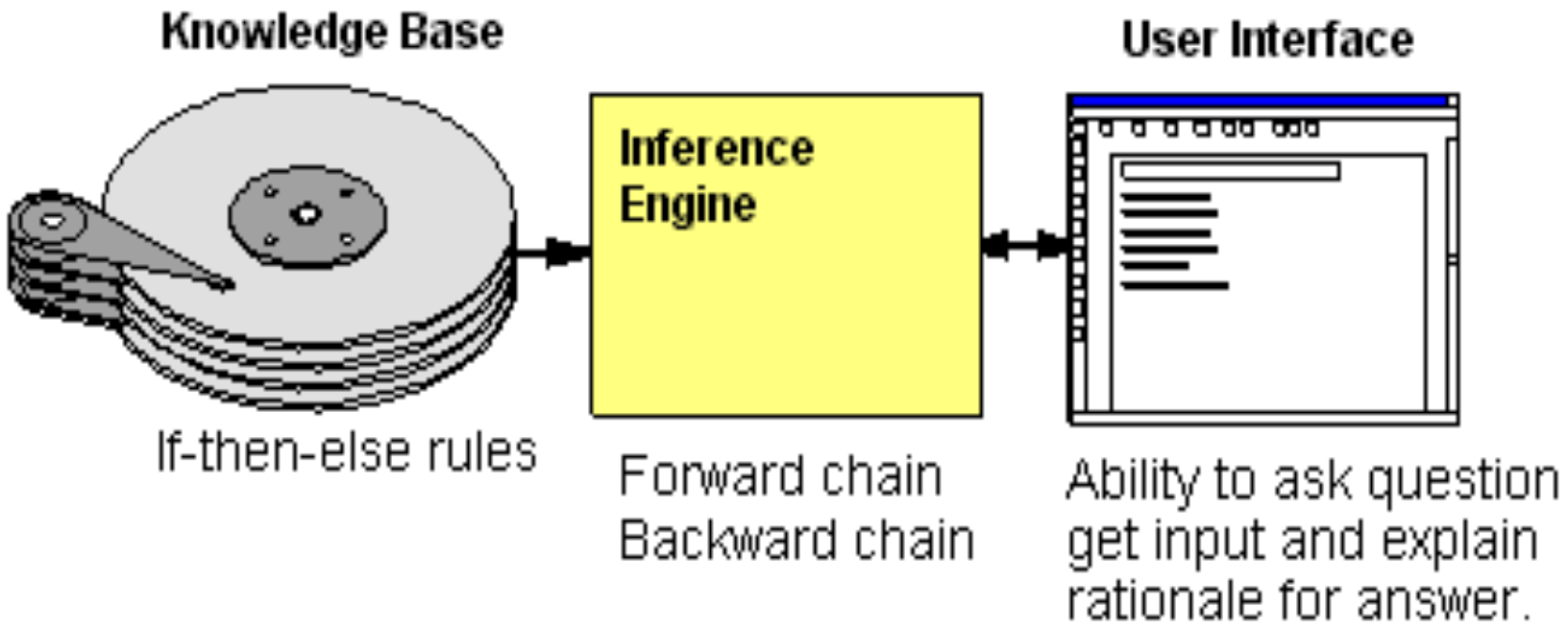


Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Image reference NVIDIA

AI Late 80s: Expert System rules engine

The definition of AI is continuously redefined : 1980s:



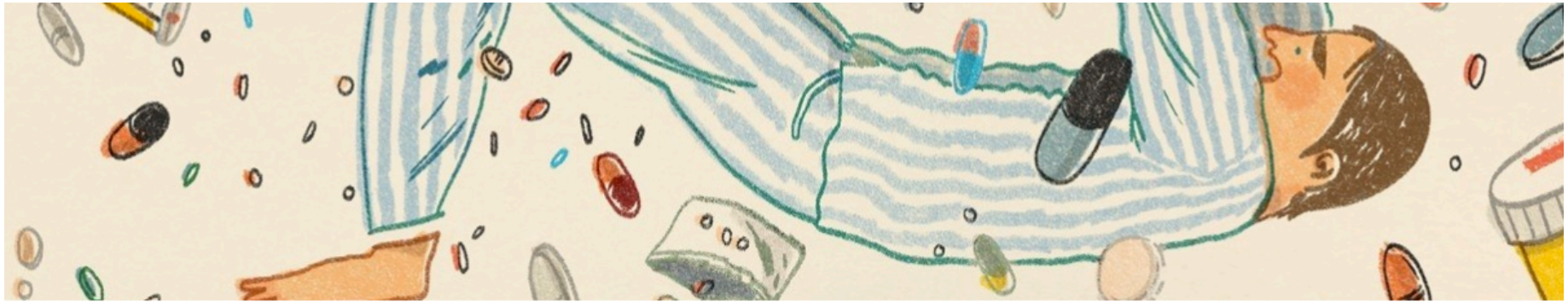
Problems with hard coded Rules

Rules are manual, uses a human expert

- **difficult to maintain**
- give a **one size fits all** decision! (2 times overdose same as 38 times)

Machine learning uses data and statistics

- can give finer grained predictions



NG

BOB WACHTER BACKCHANNEL 03.30.15 12:00 AM

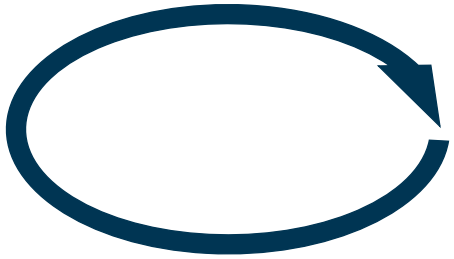
HOW TECHNOLOGY LED A HOSPITAL TO GIVE A PATIENT 38 TIMES HIS DOSAGE

What is Machine Learning?

Contains patterns

Data

Finds patterns

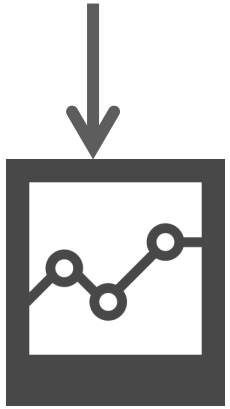


Train Algorithm

Recognizes patterns

Build Model

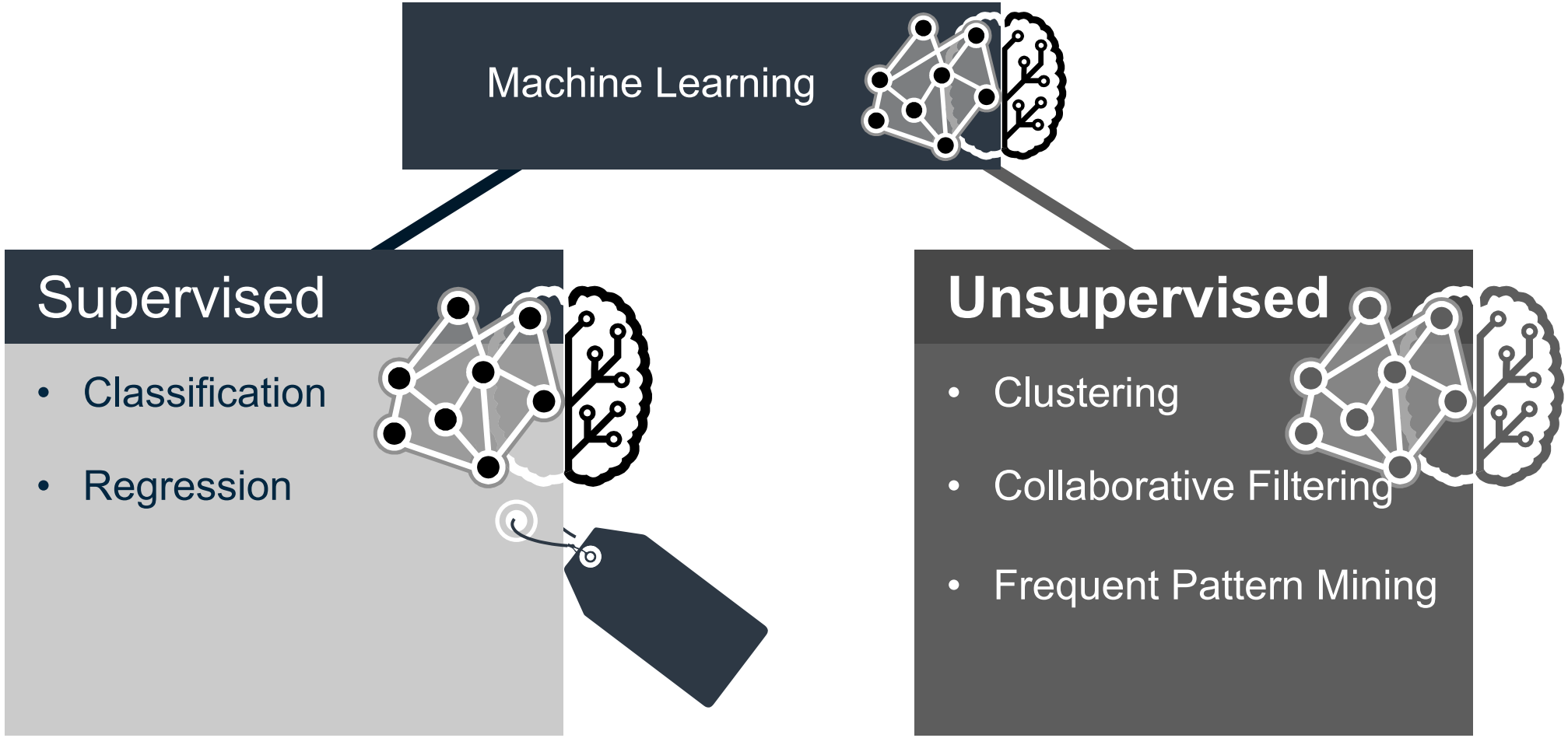
New Data



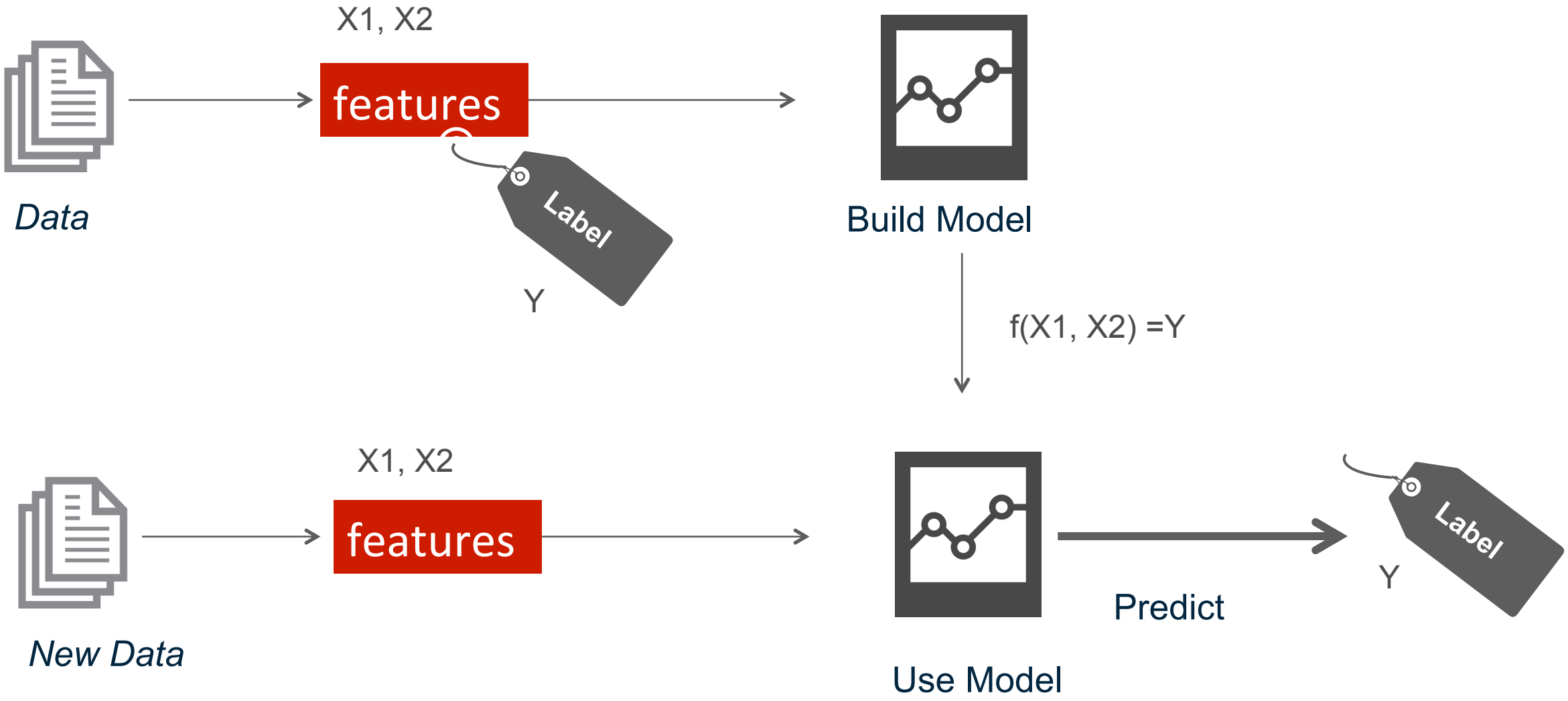
Use Model
(prediction function)

Predictions

What is Supervised Machine Learning?

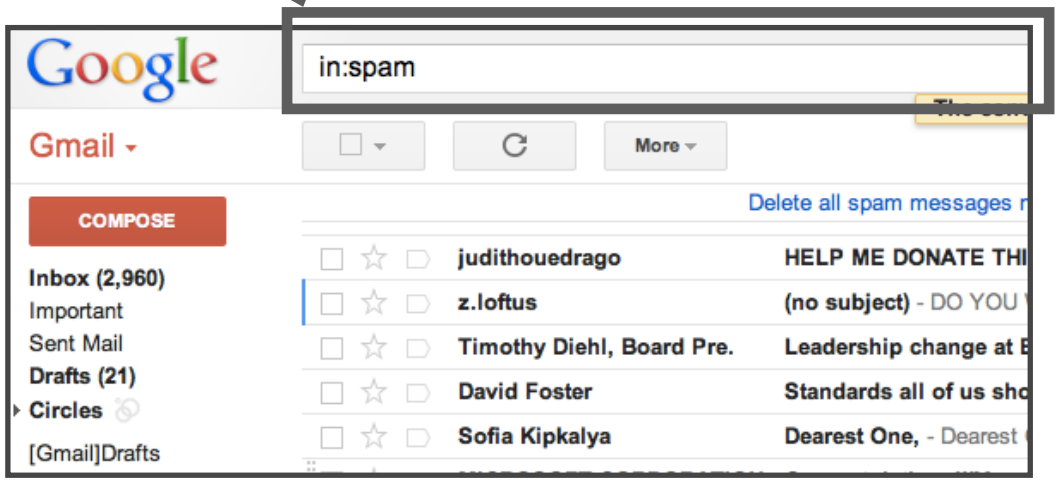


Supervised Algorithms use labeled data



Supervised Machine Learning: Classification & Regression

Classification



Identifies category for item

Classification: Definition

Form of ML that:

- Identifies which category an item belongs to
- Uses supervised learning algorithms
  - Data is labeled

If it Walks/Swims/Quacks Like a Duck …… Then It Must Be a Duck

Features:

walks

swims

quacks

ducks

Features:

walks

quacks

swims

not ducks

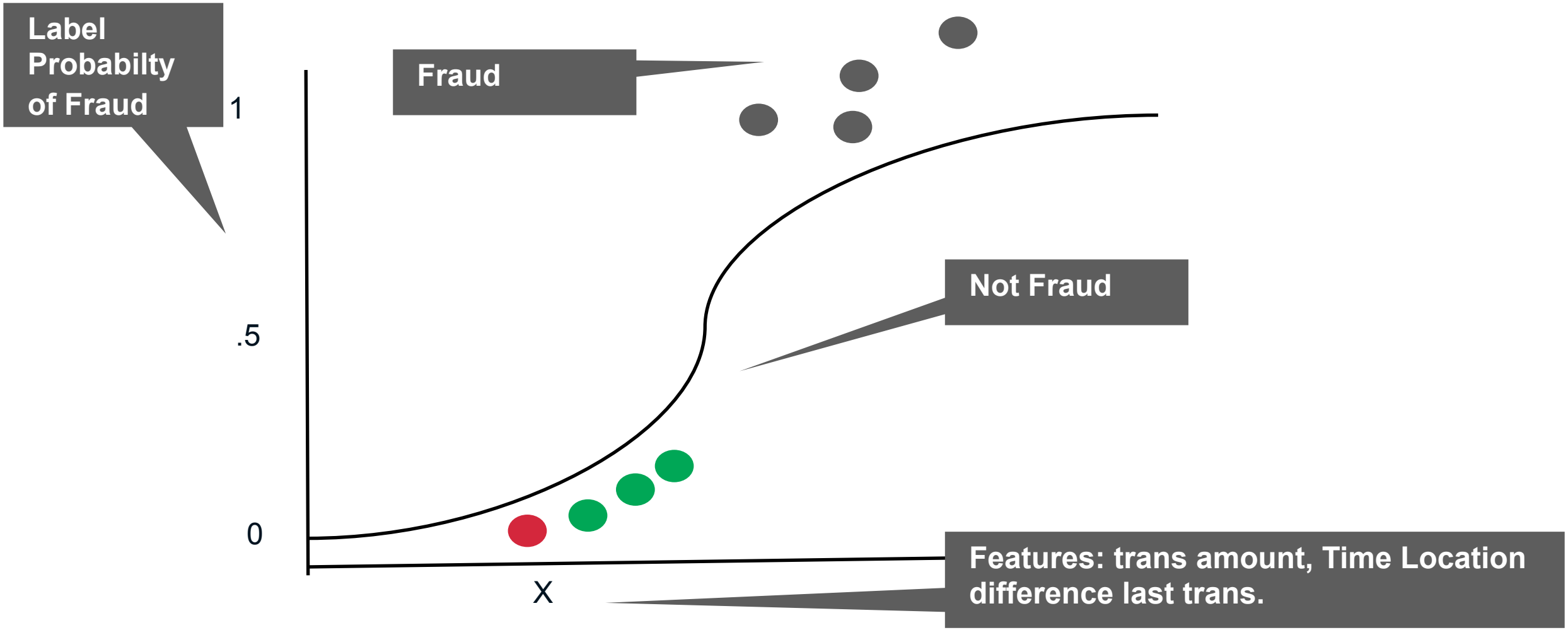
Credit Card Fraud Example

- What are we trying to predict?
  - This is the **Label**:
  - The probability of **Fraud**
- What are the "if questions" or properties we can use to predict?
  - These are the **Features**:
  - Amount $$ 24 hours > historical use ?

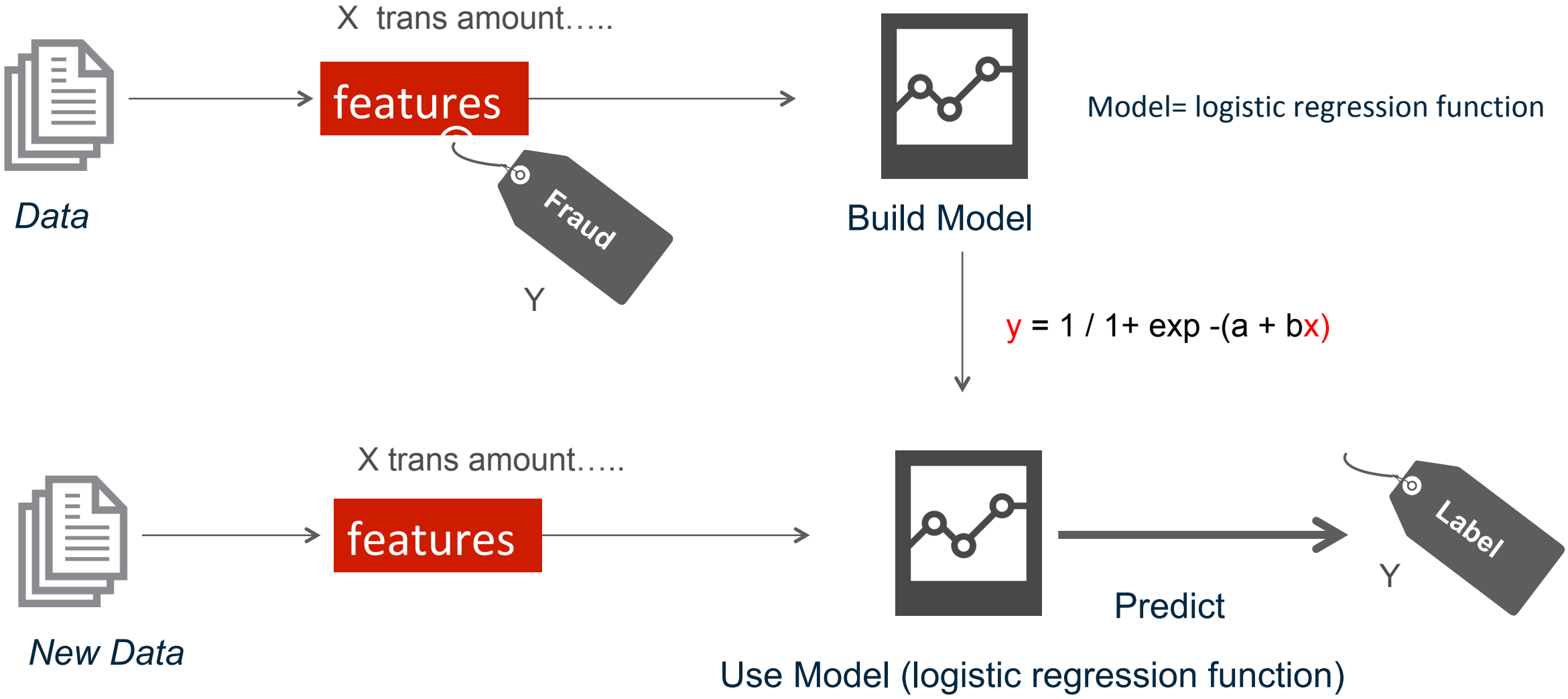
Credit Card Transaction Features
Number of Transactions last 24 hours
Total $ Amount last 24 hours
Average Amount last 24 hours
Average Amount last 24 hours compared to historical use
Location and Time difference since Last Transaction
Average transaction
fraud risk of merchant type
Merchant types for day compared to historical use

Features derived From Transaction History

Credit Card Fraud Logistic Regression Example



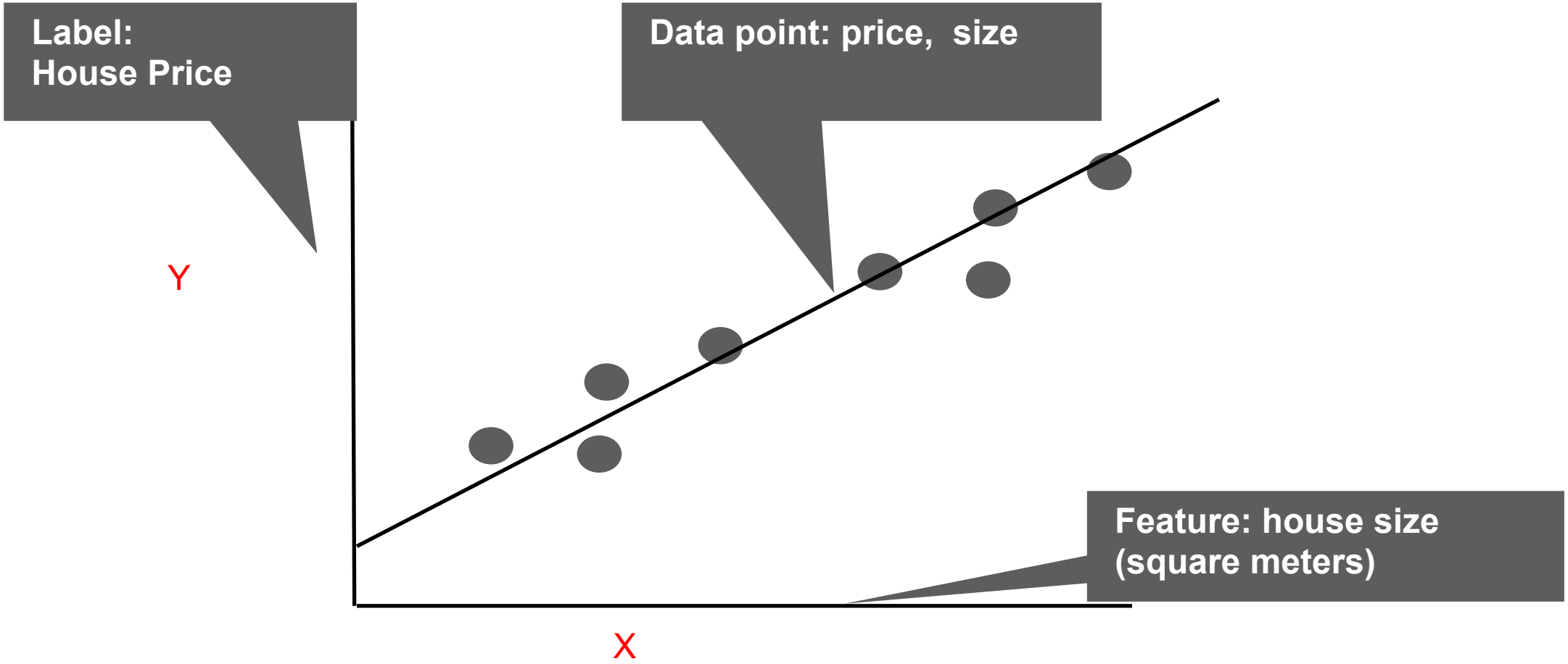
Supervised Algorithms use labeled data



House Price Prediction Example

- What are we trying to predict?
  - This is the **Label or Target outcome**:
  - The house price $
- What are the “if questions” or properties we can use to predict?
  - These are the **Features**:
  - The size of the house (square meters)

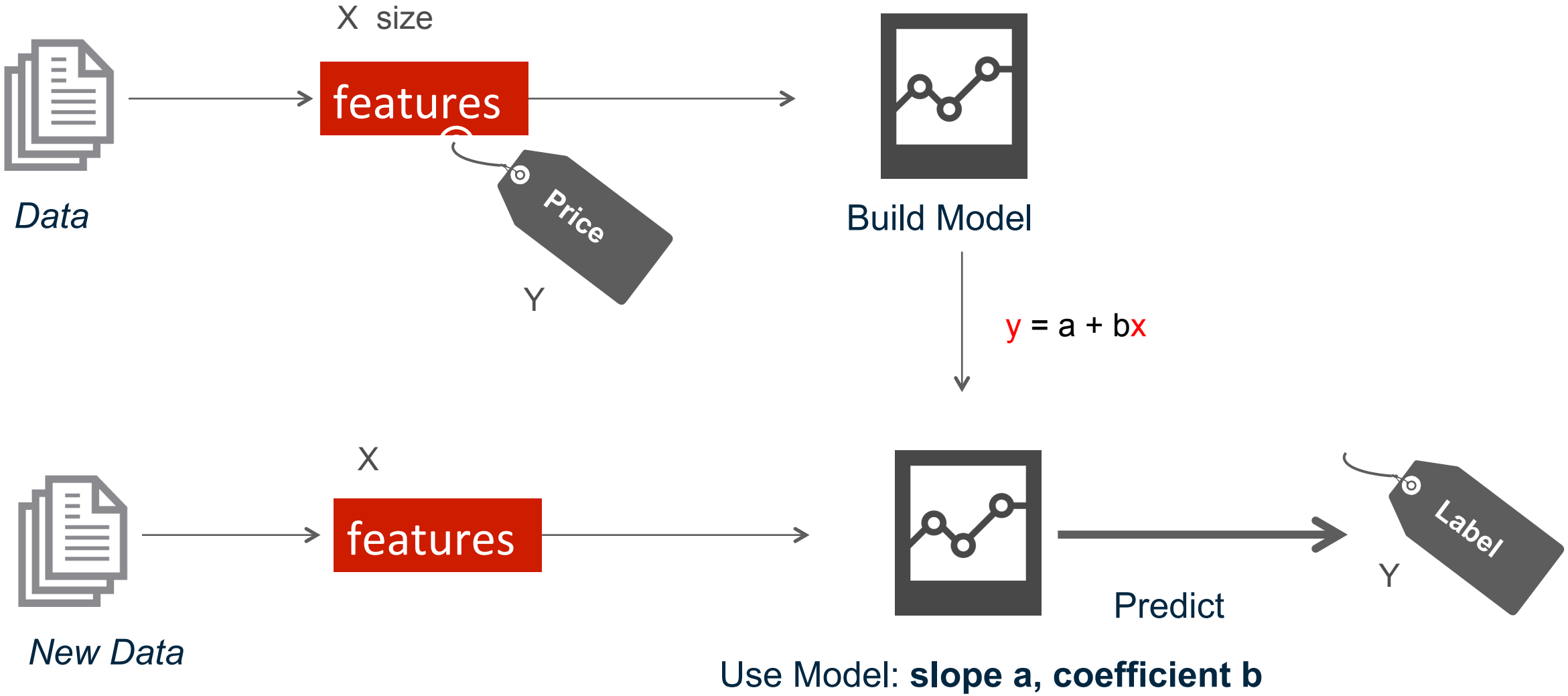
House Price Regression Example



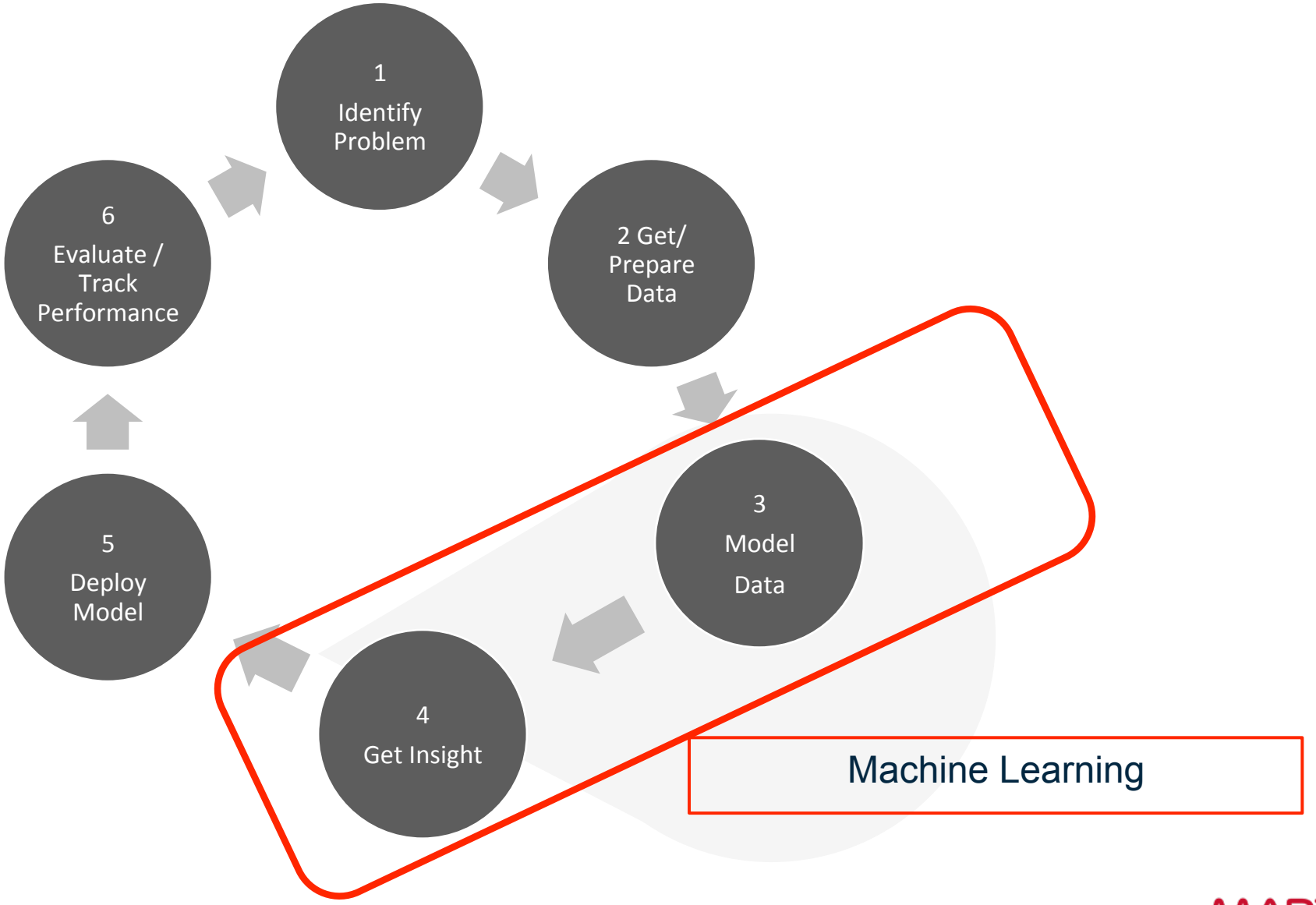
House Price = intercept + coefficient * house size

$y = a + bx$

Supervised Algorithms use labeled data



Steps to Build a Machine Learning Solution



Supervised Learning: Classification & Regression

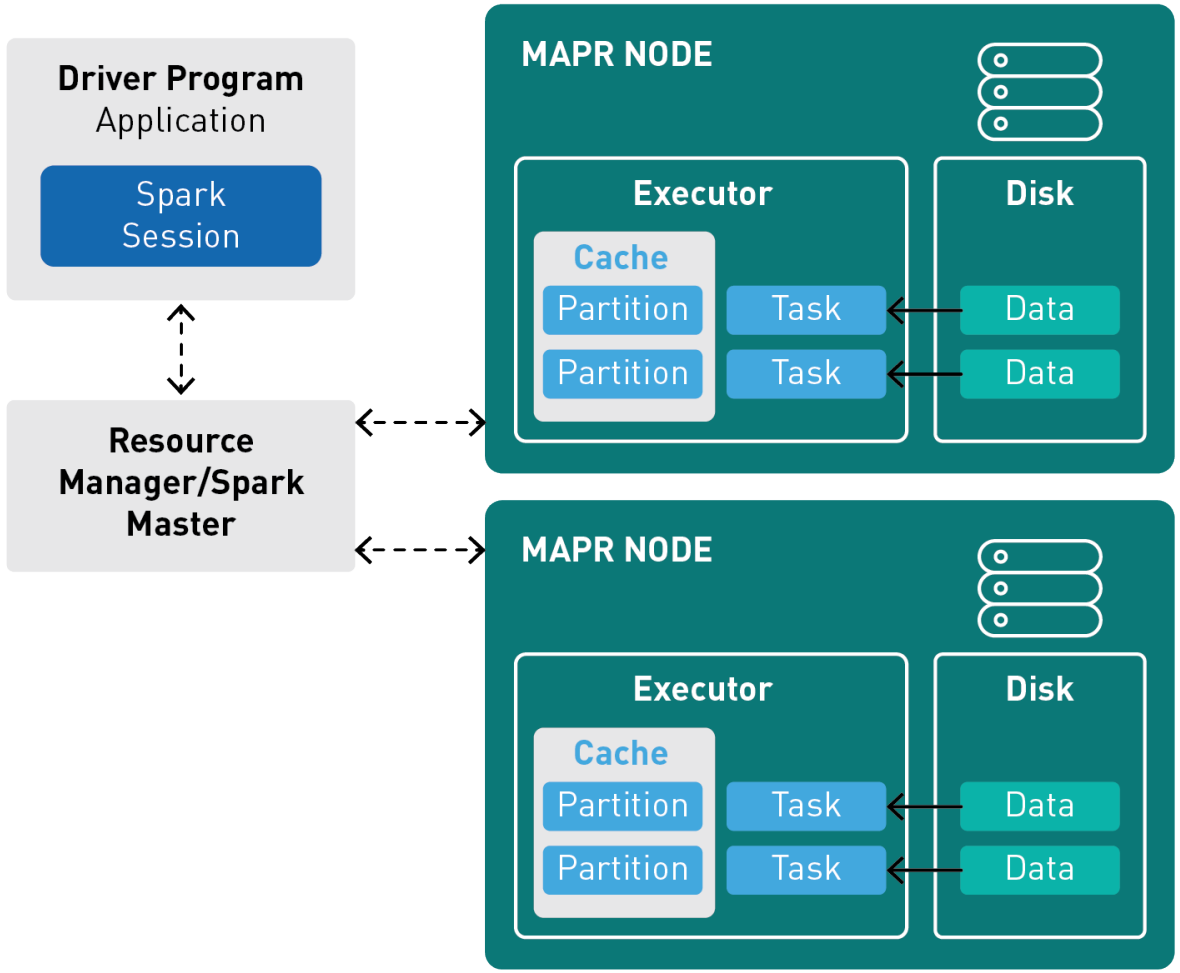
- Classification:
  - identifies which **category** (eg fraud or not fraud)
- Linear Regression:
  - predicts a **value** (eg amount of fraud)

Examples

- Retail Example:
  - Predict price, sales
- Telecom:
  - Predict customer will churn
- Healthcare Example:
  - Probability of readmission
- Marketing
  - Predict probability customer will click on add

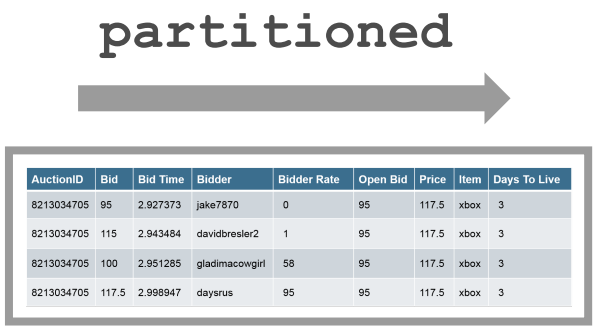
Exploring the Flight Delay Dataset

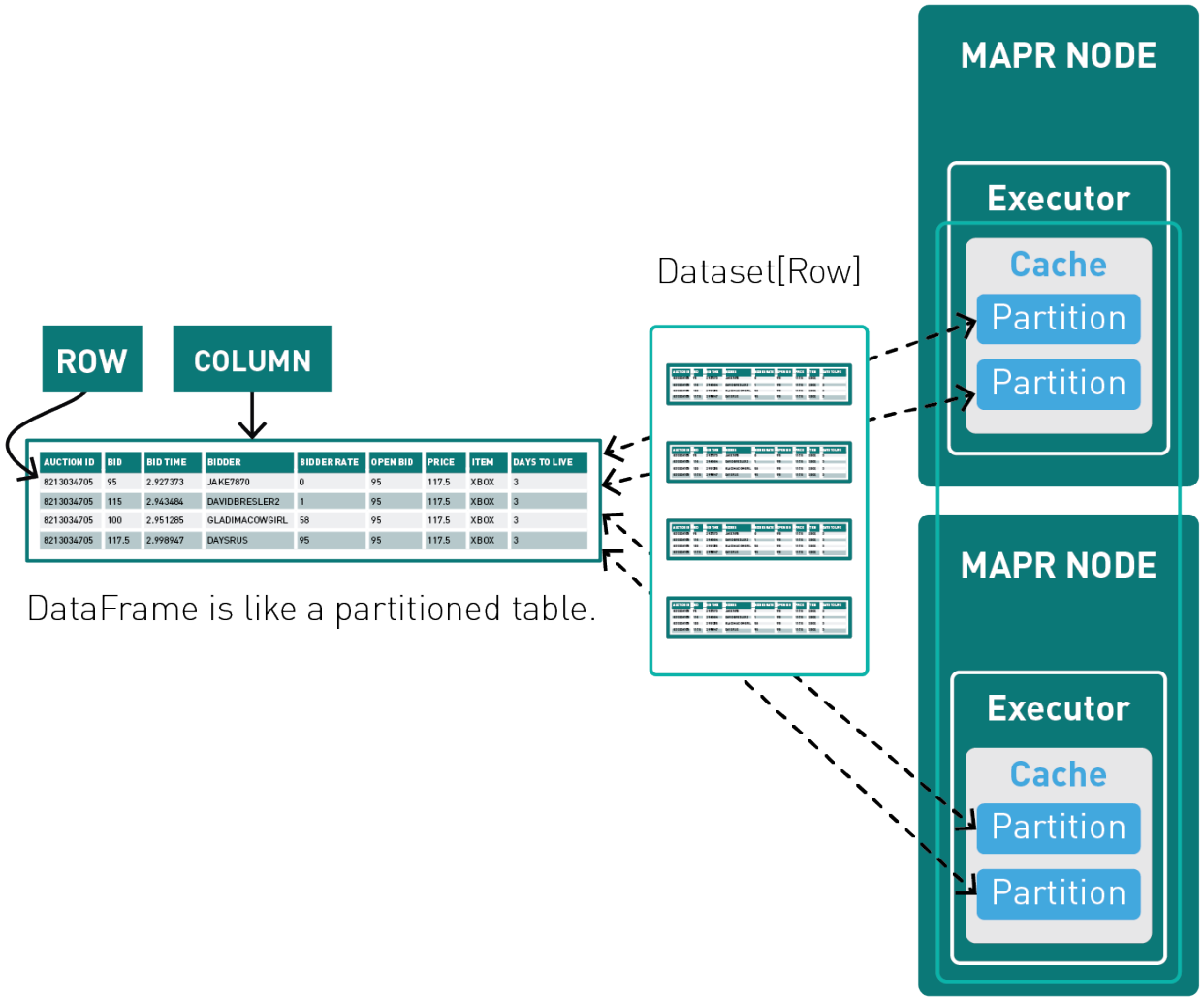
How a Spark Application Runs on a Cluster



Spark Distributed Datasets

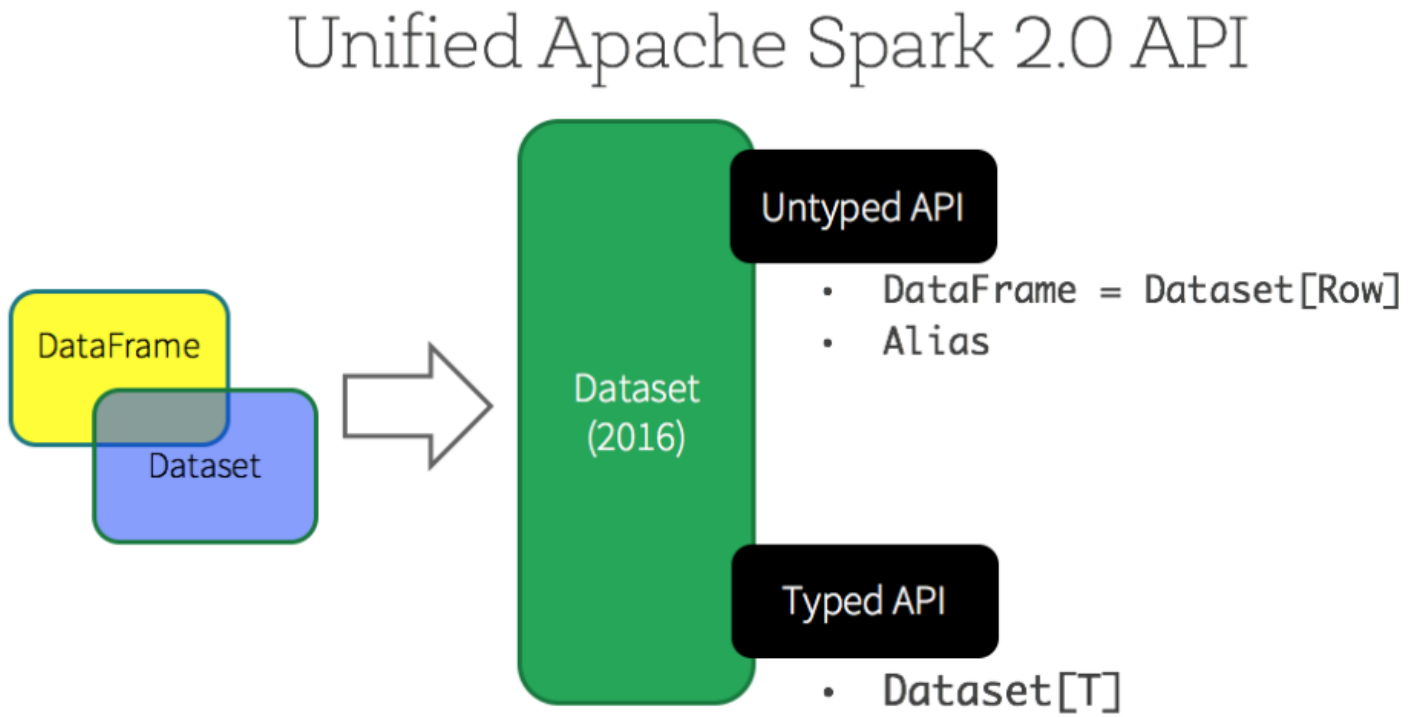
- Read only **collection** of **typed objects Dataset[T]**
- **Partitioned** across a cluster
- Operated on in **parallel**
- in memory can be **Cached**





Dataset merged with Dataframe

- in Spark 2.0, DataFrame APIs merged with Datasets APIs
- A Dataset is a collection of typed objects (SQL and functions)
    - Dataset[T]
- A DataFrame is a Dataset of generic Row objects (SQL)
    - Dataset[Row]

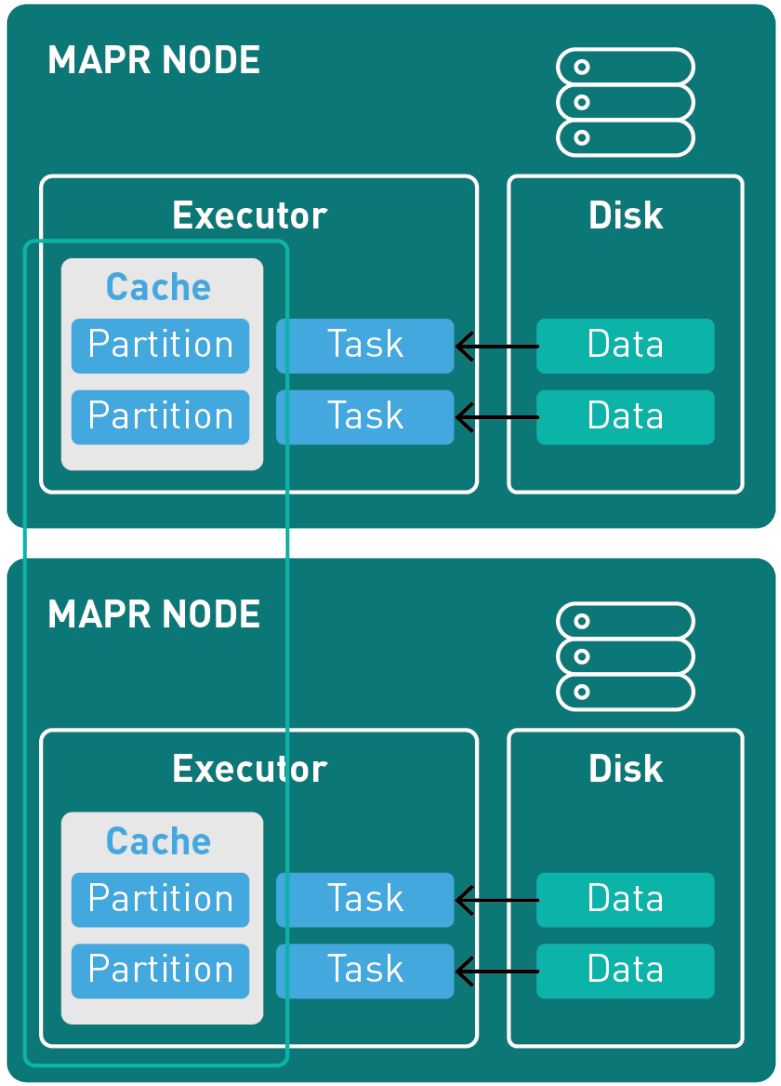


Flight Dataset

Field	Description	Example Value
id	Unique identifier: composed of carrier code, date, origin, destination, flight number	AA_2017-02-22_SFO_ORD_150
dofW (Integer)	Day of week (1=Monday 7=Sunday)	1
carrier (String)	Carrier code	AA
origin(String)	Origin Airport Code	JFK
dest(String	Destination airport code	LAX
crsdephour(Integer)	Scheduled departure hour	9
crsdeptime(Double)	Scheduled departure time	900.0
depdelay (Double)	Departure delay in minutes	40.0
crsarrtime (Double)	Scheduled arrival time	1230.0
arrdelay (Double)	Arrival delay minutes	40.0
crselapsedtime (Double)	Elapsed time	390.0
dist (Double)	Distance	2475.0


```
{
   "_id": ”ATL_LGA_2017-01-01_AA_1678",
   "dofW": 7,
   "carrier": "AA",
   "origin": "ATL",
   "dest": "LGA",
   "crsdephour": 17,
   "crsdeptime": 1700,
   "depdelay": 0.0,
   "crsarrtime": 1912,
   "arrdelay": 0.0,
   "crselapsedtime": 132.0,
   "dist": 762.0
}
```

Loading a Dataset



Loading data from a distributed file into a Dataset

Load the data into a Dataset: Define the Schema

```
case class Flight(_id: String, dofW: Integer, carrier: String, origin: String,
   dest: String,crsdephour:Integer,crsdeptime:Double,depdelay:Double, crsarrtime:Double,
   arrdelay: Double, crselapsedtime: Double, dist: Double) extends Serializable

  val schema = StructType(Array(
    StructField("_id", StringType, true),
    StructField("dofW", IntegerType, true),
    StructField("carrier", StringType, true),
    StructField("origin", StringType, true),
    StructField("dest", StringType, true),
    StructField("crsdephour", IntegerType, true),
    StructField("crsdeptime", DoubleType, true),
    StructField("depdelay", DoubleType, true),
    StructField("crsarrtime", DoubleType, true),
    StructField("arrdelay", DoubleType, true),
    StructField("crselapsedtime", DoubleType, true),
    StructField("dist", DoubleType, true)
  ))
```

Load the data into a Dataset

Load data → **Dataset**

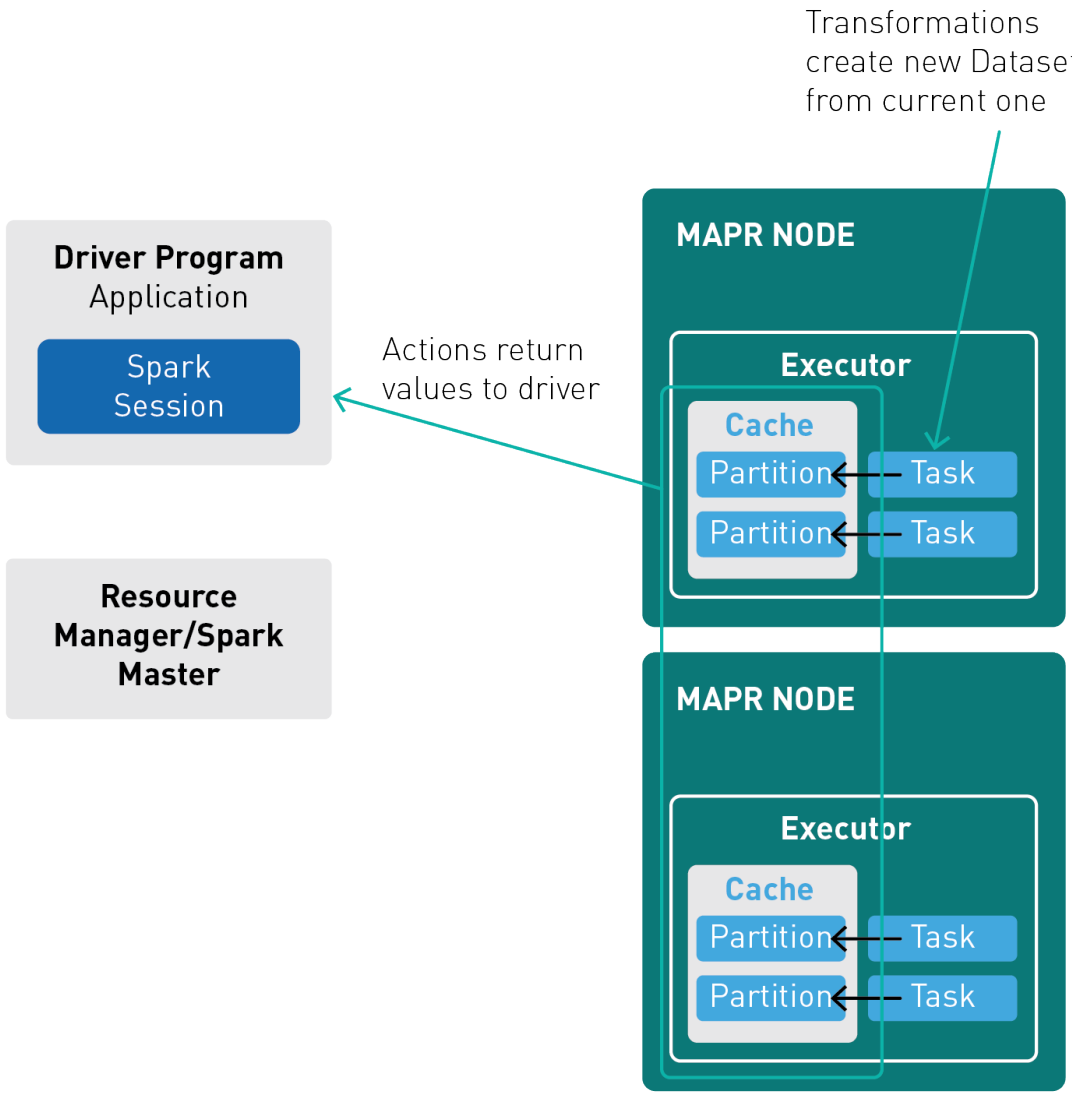
```
val df: Dataset[Flight] = spark.read.option("inferSchema", "false")
      .schema(schema).json("/p/file.json").as[Flight]
```

columns

row

```
df.show
+--------------------+----+------+------+----+---------+---------+-------+---------+-------+--------------+------+
|                 _id|dofW|carrier|origin|dest|crsdephour|crsdeptime|depdelay|crsarrtime|arrdelay|crselapsedtime|  dist|
+--------------------+----+------+------+----+---------+---------+-------+---------+-------+--------------+------+
|AA_2017-01-01_ATL...|   7|    AA|   ATL| LGA|       17|   1700.0|    0.0|   1912.0|    0.0|         132.0| 762.0|
|AA_2017-01-01_LGA...|   7|    AA|   LGA| ATL|       13|   1343.0|    0.0|   1620.0|    0.0|         157.0| 762.0|
|AA_2017-01-01_MIA...|   7|    AA|   MIA| ATL|        9|    939.0|    0.0|   1137.0|   10.0|         118.0| 594.0|
|AA_2017-01-01_ORD...|   7|    AA|   ORD| MIA|       20|   2020.0|    0.0|     26.0|    0.0|         186.0|1197.0|
```

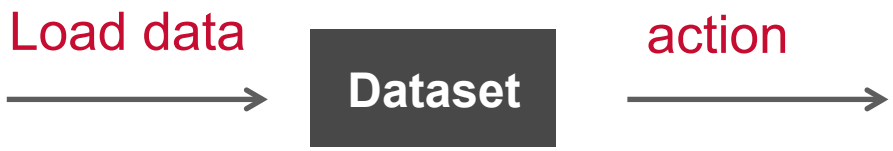
Spark Distributed Datasets



Transformations create a new Dataset from the current one,
Lazily evaluated

Actions return a value to the driver

Action on a Dataset



df.take(1)

result:

Array[Flight] = Array(Flight(ATL_LGA_2017-01-01_17_AA_1678,
7,AA,ATL,LGA,17,1700.0,0.0,1912.0,0.0,132.0,762.0))

DataFrame Transformations

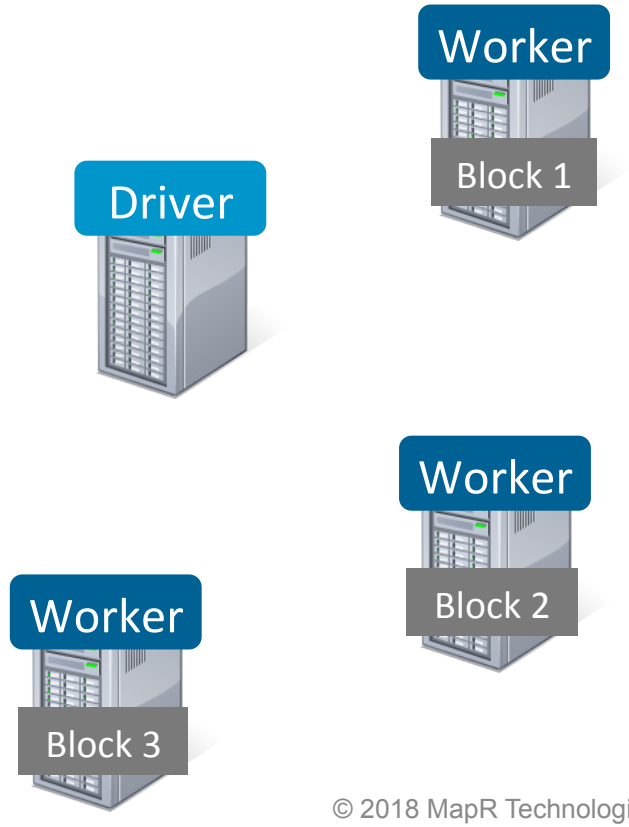
L I Query	Description
`agg(expr, exprs)`	Aggregates on entire DataFrame
`distinct`	Returns new DataFrame with unique rows
`except(other)`	Returns new DataFrame with rows from this DataFrame not in other DataFrame
`filter(expr); where(condition)`	Filter based on the SQL expression or condition
`groupBy(cols: Columns)`	Groups DataFrame using specified columns
`join (DataFrame, joinExpr)`	Joins with another DataFrame using given join expression
`sort(sortcol)`	Returns new DataFrame sorted by specified column
`select(col)`	Selects set of columns

DataFrame Actions

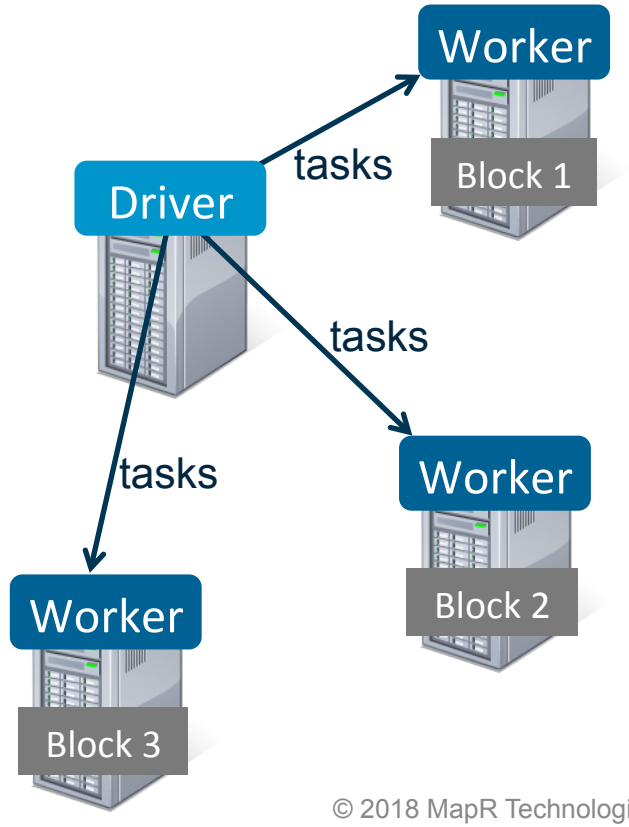
Action	Description
`collect()`	Returns array containing all rows in DataFrame
`count()`	Returns number of rows in DataFrame
`describe(cols:String*)`	Computes statistics for numeric columns (count, mean, stddev, min and max)
`first()`	Returns the first row
`head()`	Returns the first row
`show()`	Displays first 20 rows
`take(n:int)`	Returns the first n rows

Example loading a Dataset

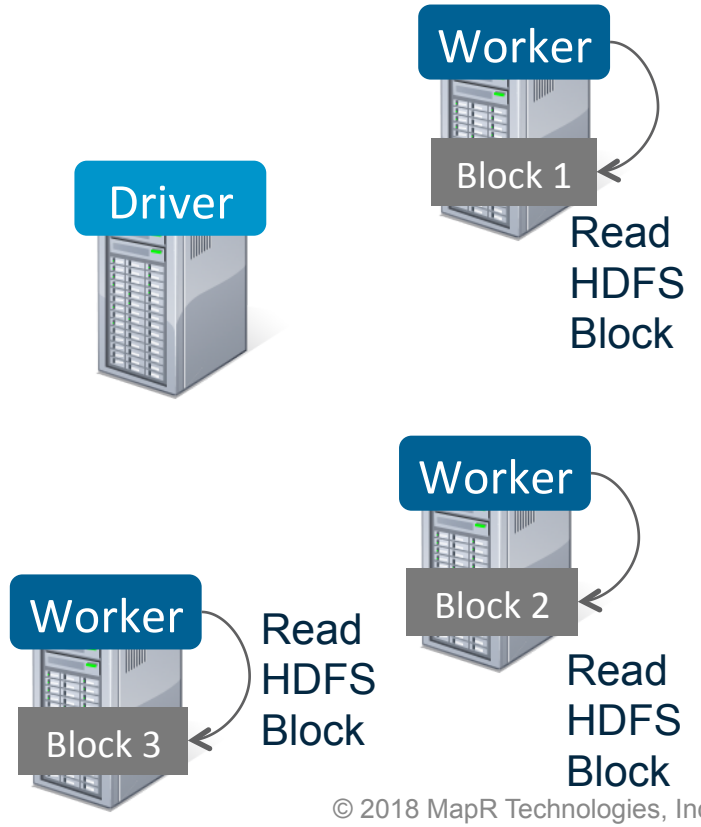
```
val df: Dataset[Flight] = spark.read.json(”/p/file.json").as[Flight]
df.cache
df.count
```



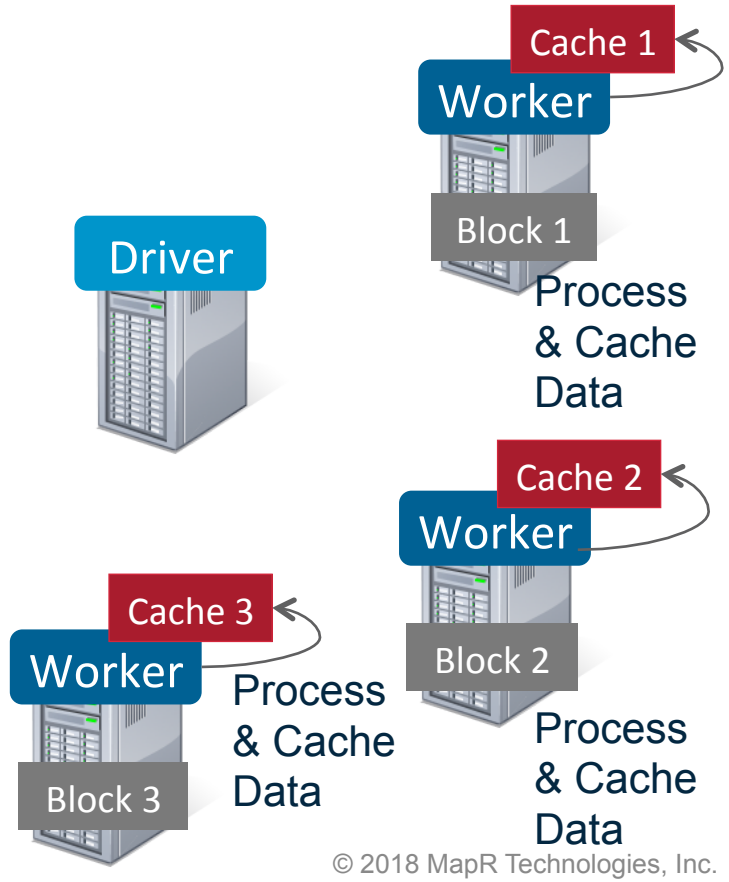
Example:



Example

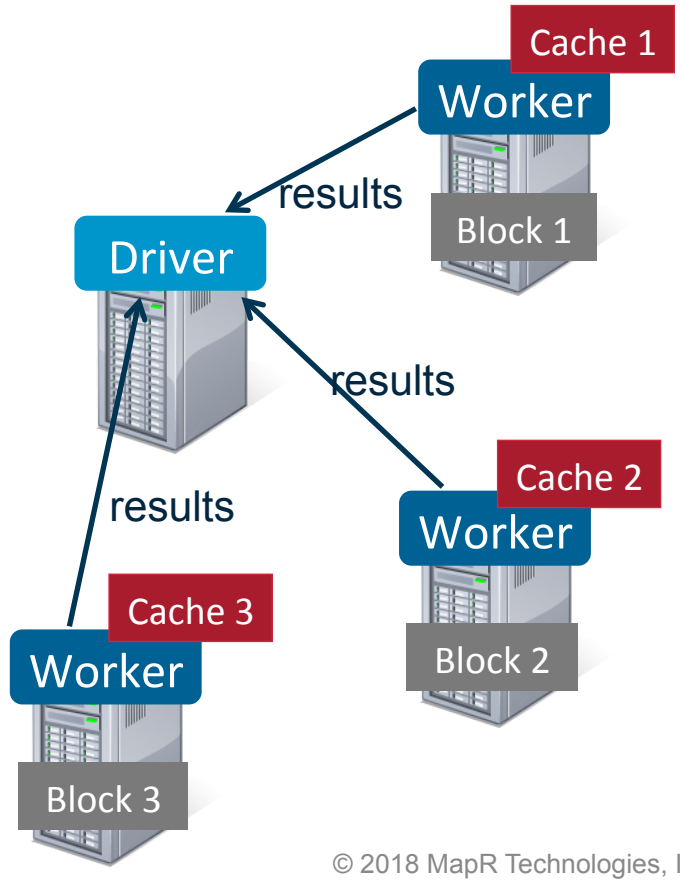


Example



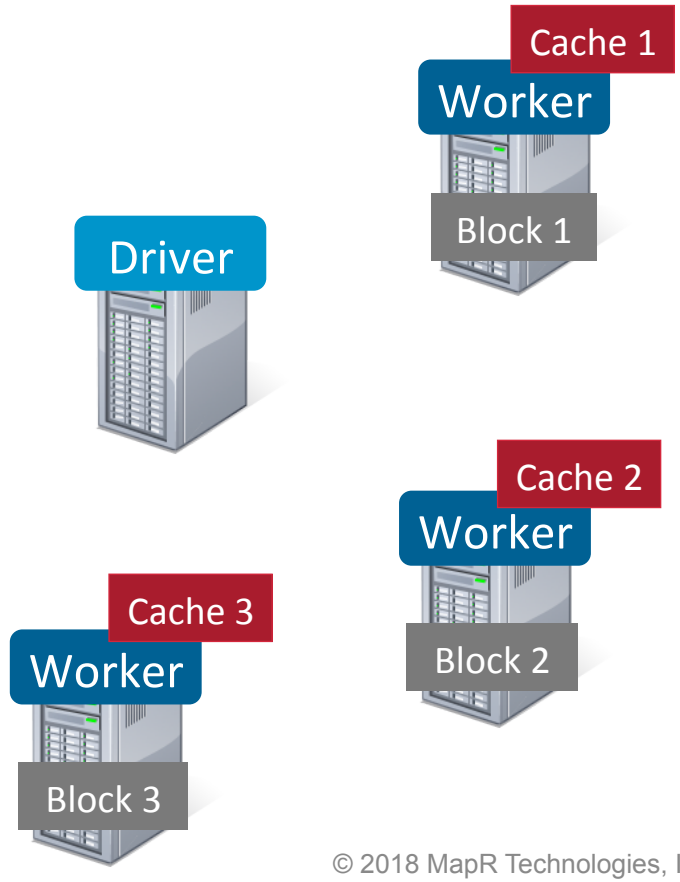
Example:

```
val df: Dataset[Flight] = spark.read.json(”/p/file.json").as[Flight]
df.cache
df.count

res9: Long = 41348
```

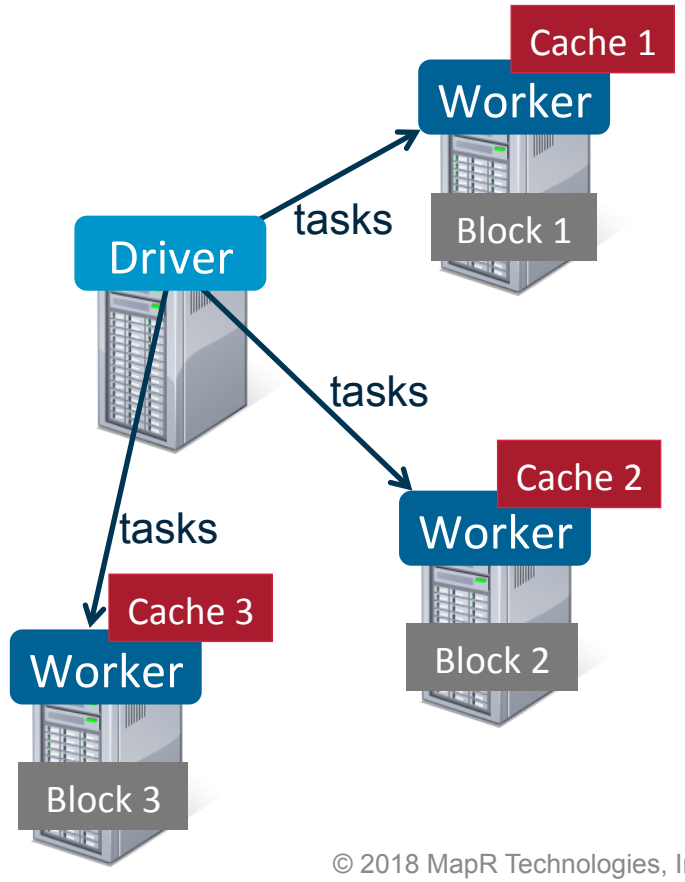


Example

```
val df: Dataset[Flight] = spark.read.json(”/p/file.json").as[Flight]
df.cache
df.count
df.show
```



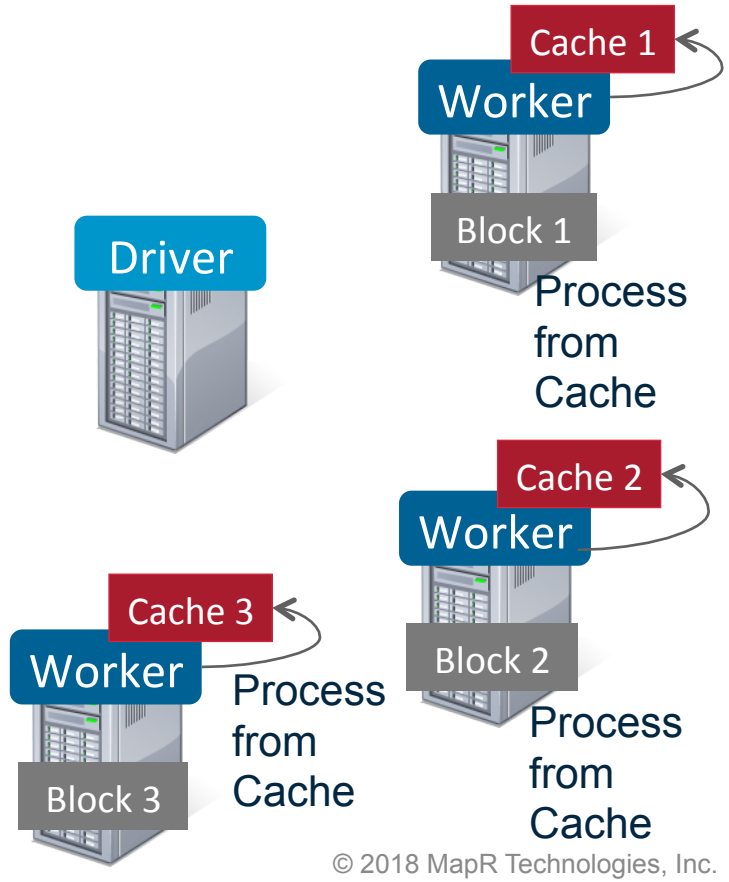
Example



Example: Log Mining

```
val df: Dataset[Flight] = spark.read.json(”/p/file.json").as[Flight]
df.cache
df.count
df.show
```

Cached, does not have to read from file again



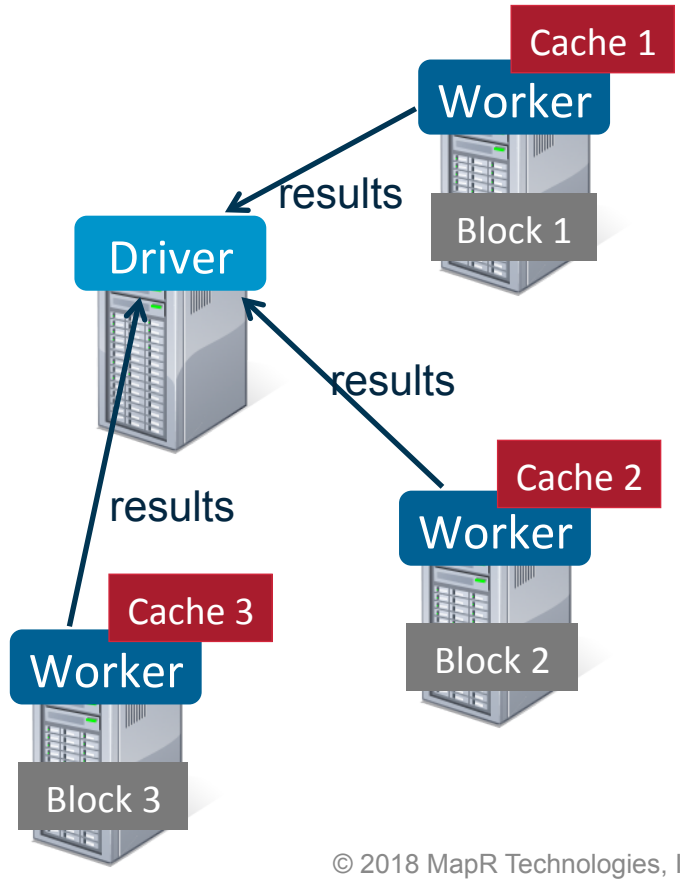
Example:

val df: Dataset[Flight] = spark.read.json(”/p/file.json").as[Flight]

```
df.cache
df.count
df.show
```

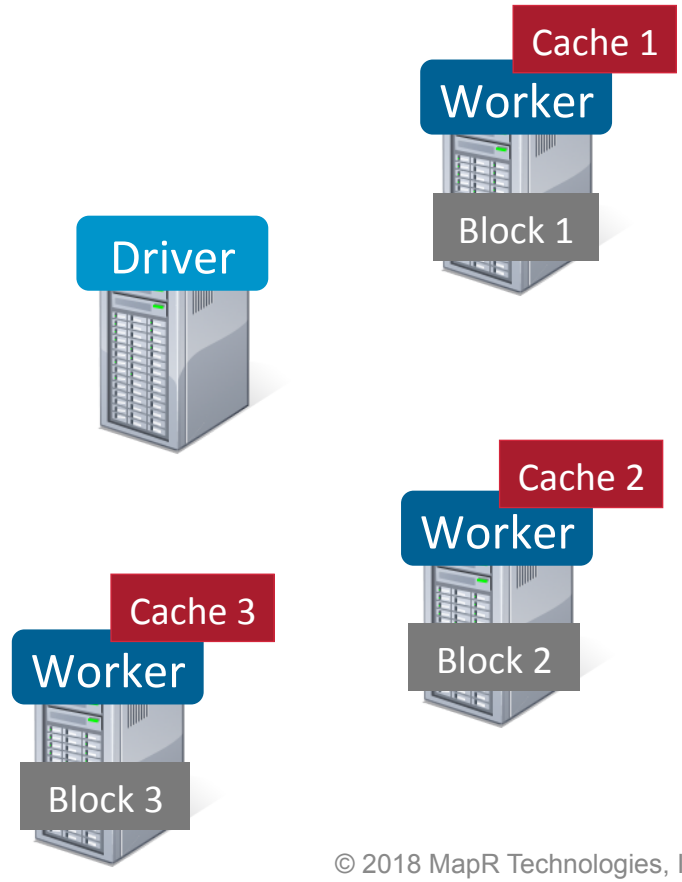
```
df.show
```

_id	dofW	carrier	origin	dest	crsdephour	crsdeptime	depdelay	crsarrtime	arrdelay	crselapsedtime	dist
AA_2017-01-01_ATL...	7	AA	ATL	LGA	17	1700.0	0.0	1912.0	0.0	132.0	762.0
AA_2017-01-01_LGA...	7	AA	LGA	ATL	13	1343.0	0.0	1620.0	0.0	157.0	762.0
AA_2017-01-01_MIA...	7	AA	MIA	ATL	9	939.0	0.0	1137.0	10.0	118.0	594.0
AA_2017-01-01_ORD...	7	AA	ORD	MIA	20	2020.0	0.0	26.0	0.0	186.0	1197.0



Example:

```
val df: Dataset[Uber] = spark.read.option("inferSchema",
   "false").schema(schema).csv("data/uber.csv").as[Uber]
df.cache
df.count
df.show
```



Cache your data ➔ Faster Results

Spark Use Cases

Iterative Algorithms on **large amounts** of data

Some Algorithms that need **iterations**

- Clustering (K-Means)
- Linear Regression
- Graph Algorithms (e.g., PageRank)
- Alternating Least Squares ALS

Some Example Use Cases:

- Anomaly detection
- Classification
- Recommendations

Destinations with highest number of Departure Delays

```
df.filter($"depdelay" > 40).groupBy(”dest”)
      .count().orderBy(desc(count)).show(10)
```

```
+----+-----+
|dest|count|
+----+-----+
| SFO|  711|
| EWR|  620|
| ORD|  593|
| ATL|  547|
| LGA|  535|
| DEN|  397|
| MIA|  385|
| BOS|  343|
| IAH|  333|
+----+-----+
```

Top 5 Longest Departure Delays

```
df.select($"carrier",$"origin",$"dest”,$"depdelay”,
 $"crsdephour”).filter($"depdelay" > 40)
 .orderBy(desc( "depdelay" )).show(5)
```

```
+-------+------+----+--------+----------+
|carrier|origin|dest|depdelay|crsdephour|
+-------+------+----+--------+----------+
|     AA|   SFO| ORD|  1440.0|         8|
|     DL|   BOS| ATL|  1185.0|        17|
|     UA|   DEN| EWR|  1138.0|        12|
|     DL|   ORD| ATL|  1087.0|        19|
|     UA|   MIA| EWR|  1072.0|        20|
+-------+------+----+--------+----------+
```

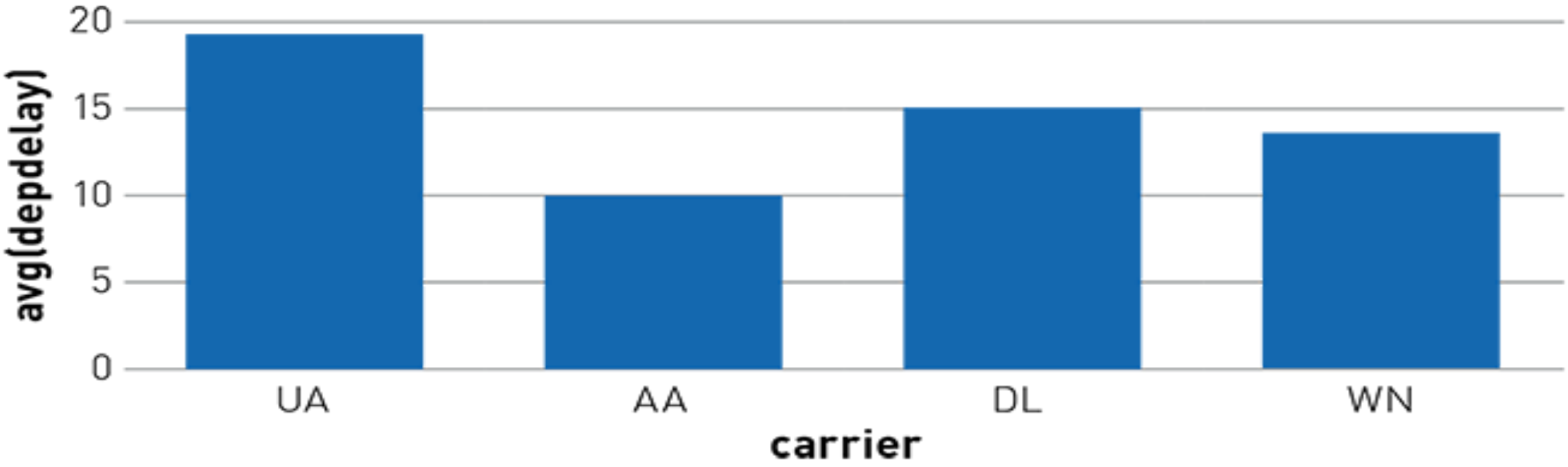
Register Dataframe as a Temporary View

Register Dataset as a Temporary View in order to explore with SQL

```
df.createOrReplaceTempView("flights")
spark.catalog.cacheTable("flights")
```

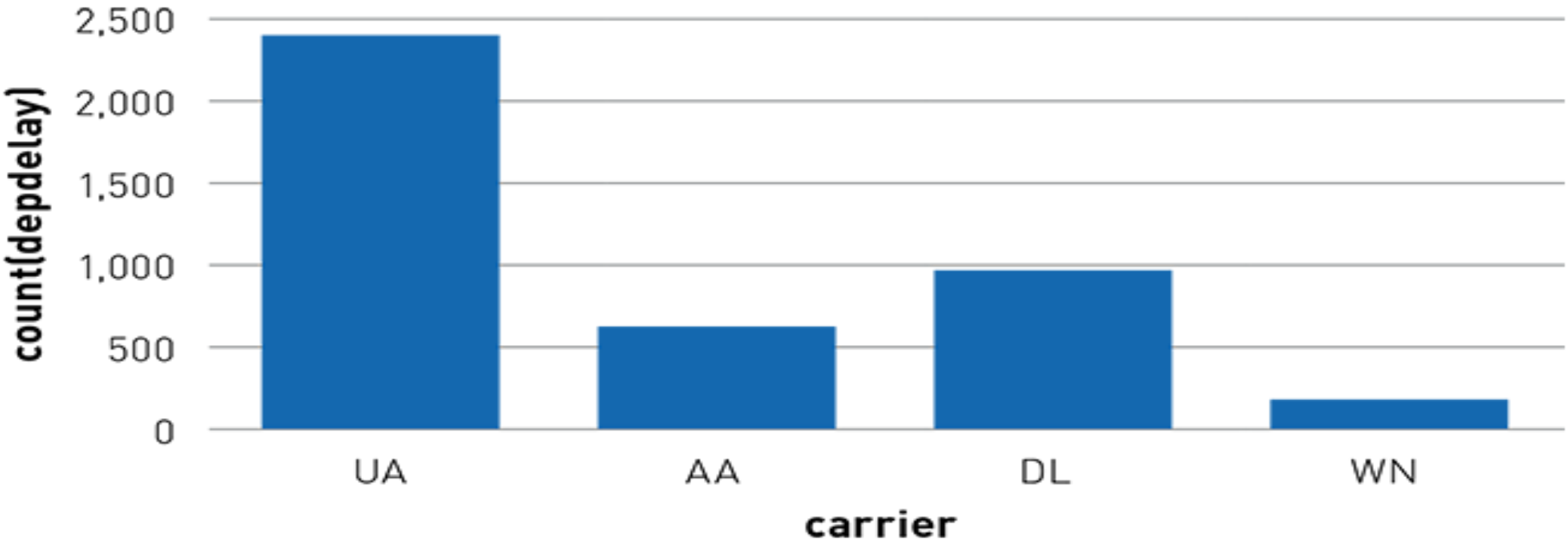
Average Departure Delay by Carrier

```
%sql select carrier, avg(depdelay) from flights
 group by carrier
```



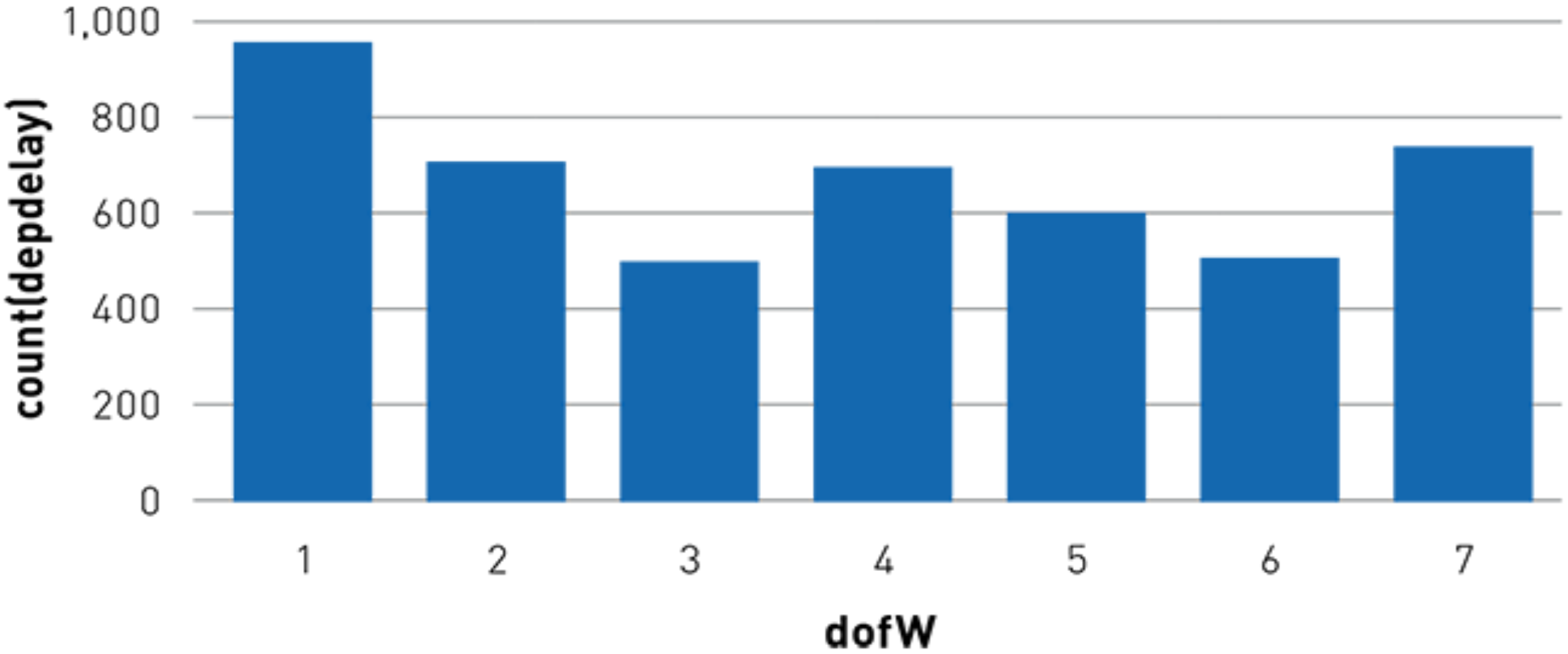
Count of Departure Delays by Carrier

```
%sql select carrier, count(depdelay) from
flights where depdelay > 40 group by carrier
```



Count of Departure Delays by Day of the Week

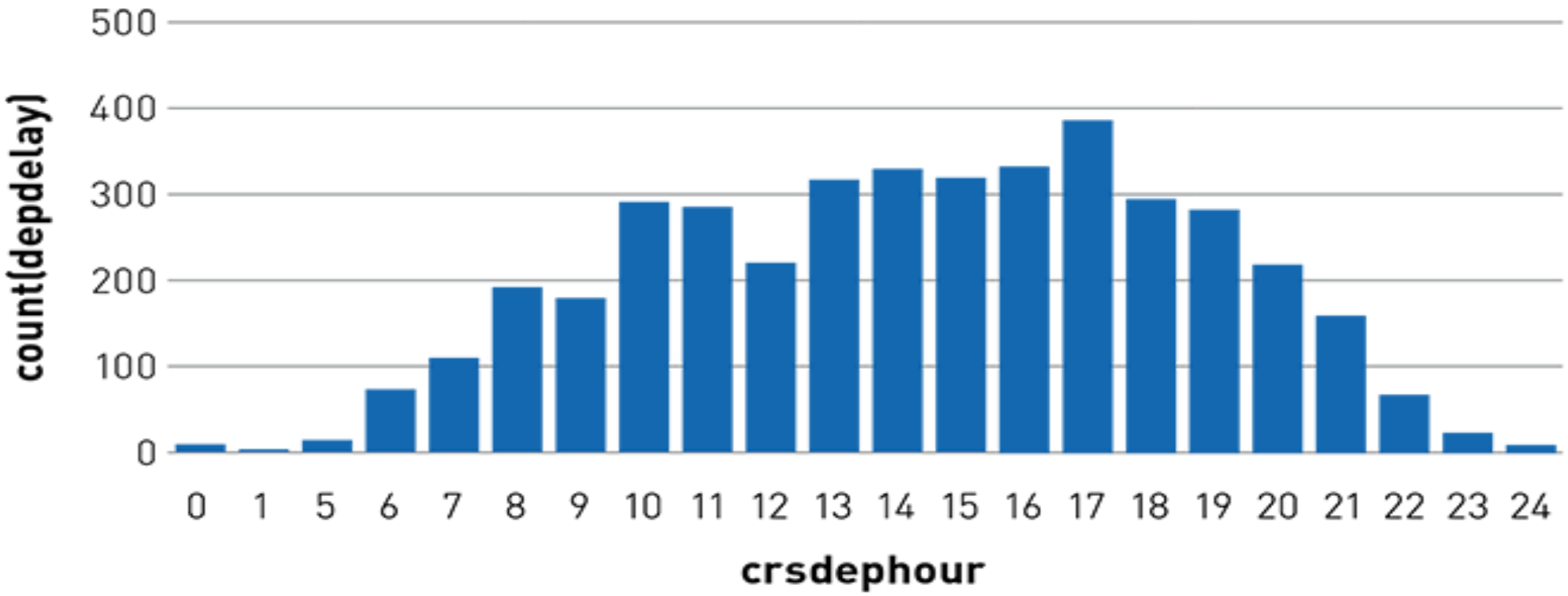
```
%sql select dofw, count(depdelay) from flights
where depdelay > 40 group by dofw
```



Monday= 1 Sunday = 7

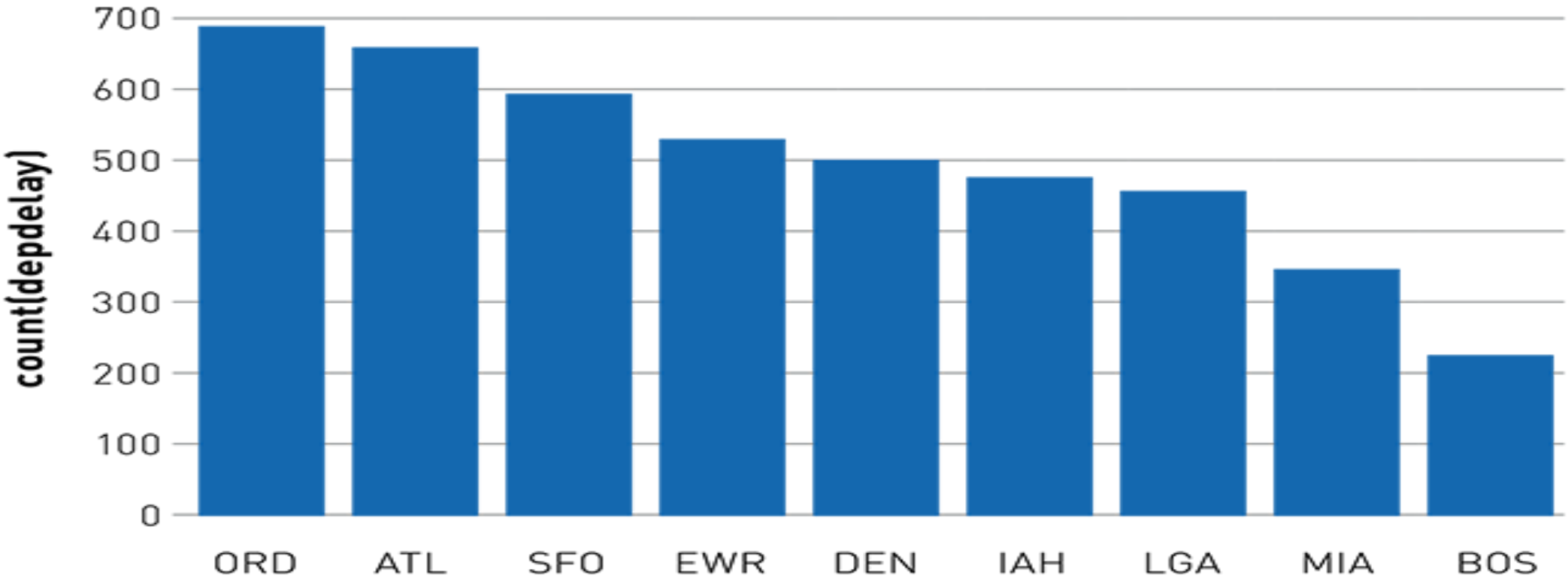
Count of Departure Delays by Hour of the Day

```
%sql select crsdephour, count(depdelay) from
flights where depdelay > 40 group by crsdephour
```



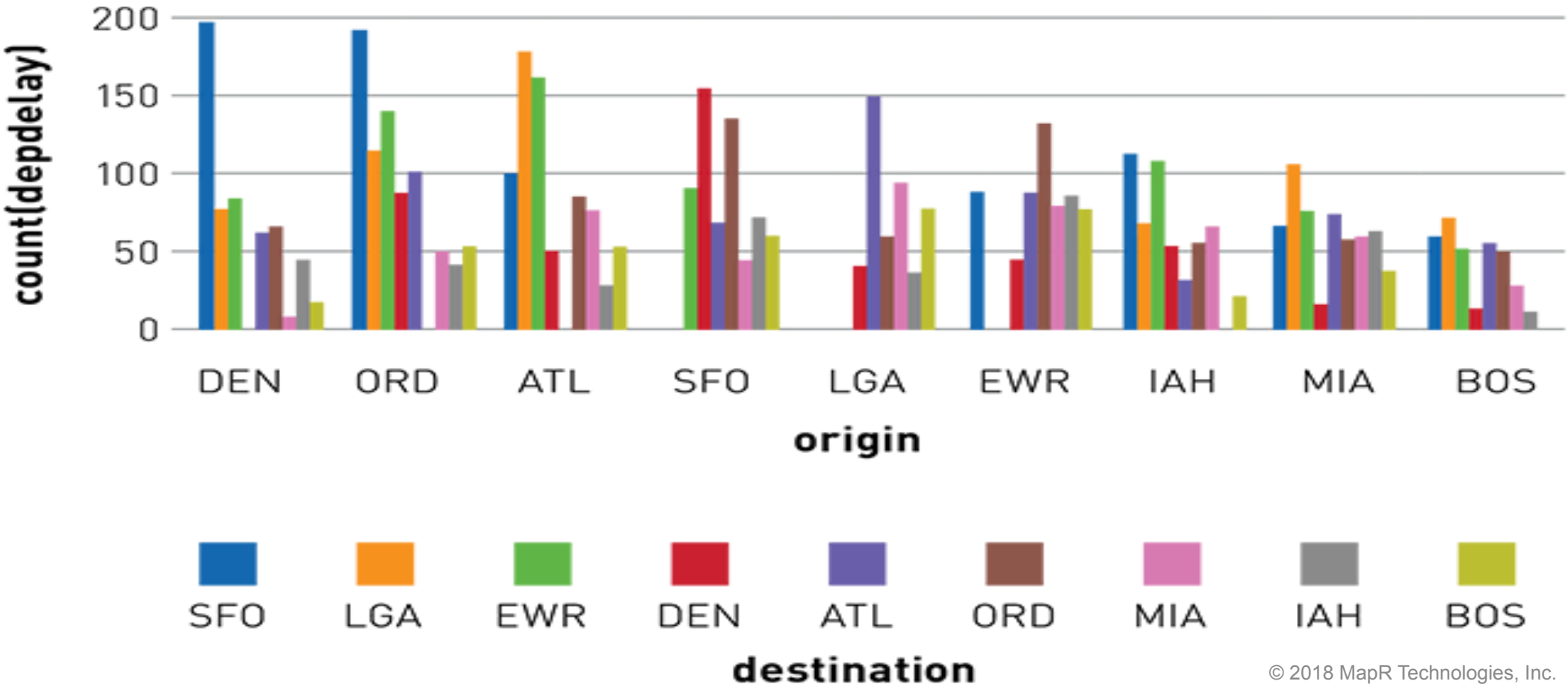
Count of Departure Delays by Origin

```
%sql select origin, count(depdelay) from flights
where depdelay > 40 group by origin
```



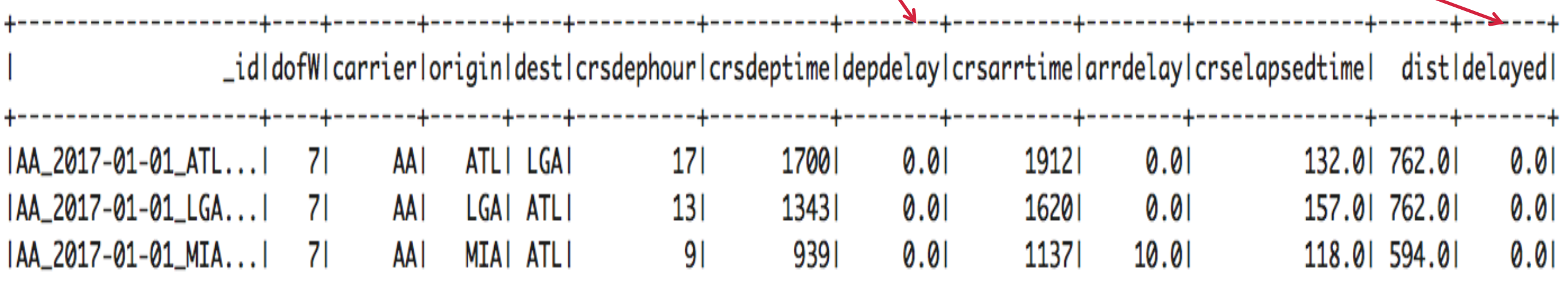
Count of Departure Delays by Origin, Destination

```
%sql select origin,dest count(depdelay) from
flights where depdelay > 40 group by origin,dest
```



Bucket Dataset for Delayed not Delayed

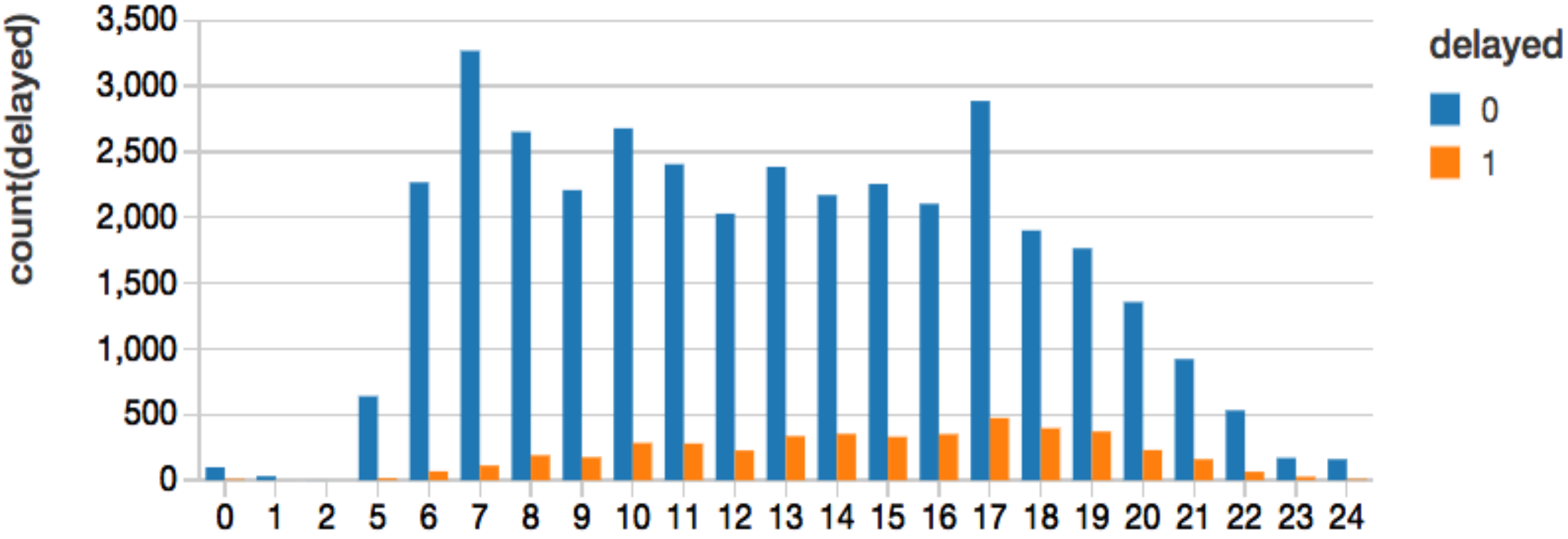
```
val delaybucketizer = new Bucketizer()
.setInputCol("depdelay”)
.setOutputCol("delayed")
.setSplits(Array( 0.0,  40.0))

val df2 = delaybucketizer.transform(df1)
```



_id	dofW	carrier	origin	dest	crsdephour	crsdeptime	depdelay	crsarrtime	arrdelay	crselapsedtime	dist	delayed
AA_2017-01-01_ATL...	7	AA	ATL	LGA	17	1700	0.0	1912	0.0	132.0	762.0	0.0
AA_2017-01-01_LGA...	7	AA	LGA	ATL	13	1343	0.0	1620	0.0	157.0	762.0	0.0
AA_2017-01-01_MIA...	7	AA	MIA	ATL	9	939	0.0	1137	10.0	118.0	594.0	0.0

Count of Departure Delays by Origin, Destination

```
%sql select origin,dest count(depdelay) from
flights where depdelay > 40 group by origin,dest
```



Bucket Dataset for Delayed not Delayed

```
df2.groupBy("delayed").count.show
```

result:

```
+-------+-----+
|delayed|count|
+-------+-----+
|    0.0|36790|
|    1.0| 4558|
+-------+-----+
```

Stratify

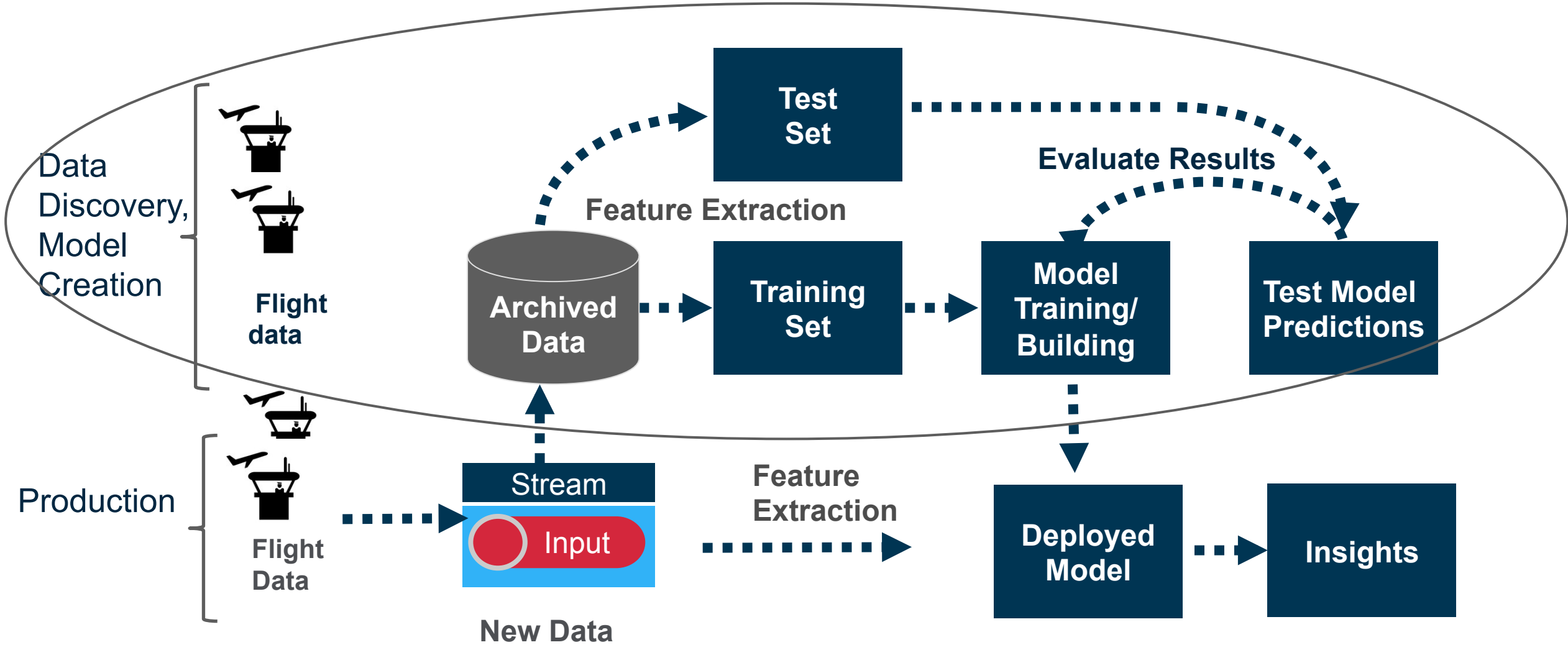
```
val fractions = Map(0.0 -> .13, 1.0 -> 1.0)
Val train = df.stat.sampleBy("delayed", fractions, 36L)

train.groupBy("delayed").count.show
```

result:

```
+-------+-----+
|delayed|count|
+-------+-----+
|    0.0| 4766|
|    1.0| 4558|
+-------+-----+
```

Predict Flight Delays

ML Discovery Model Building



Flight Delay Example

- What are we trying to predict?
  - This is the **Label or Target outcome**:
  - Delayed = 1 if delay > 40 minutes
- What are the “if questions” or properties we can use to predict?
- These are the **Features**:
  - Day of the week, scheduled departure time, scheduled arrival time, Airline carrier, scheduled travel time, Origin, Destination, Distance

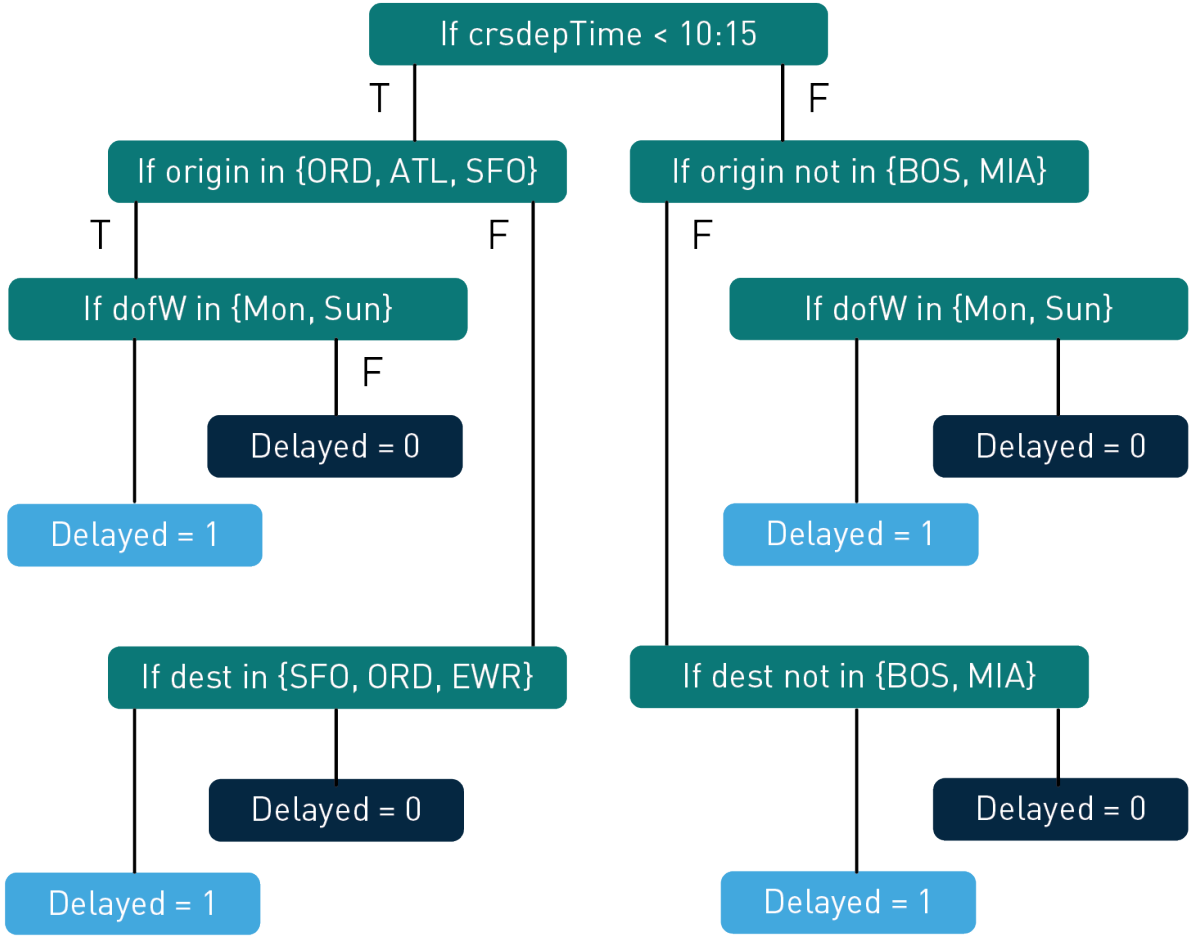
label

features

delayed	dofW	crsdepTime	crsArrTime	carrier	elapTime	orign	dest	dist
1.0/0.0	1	1015	1230	AA	385.0	JFK	LAX	2475.0

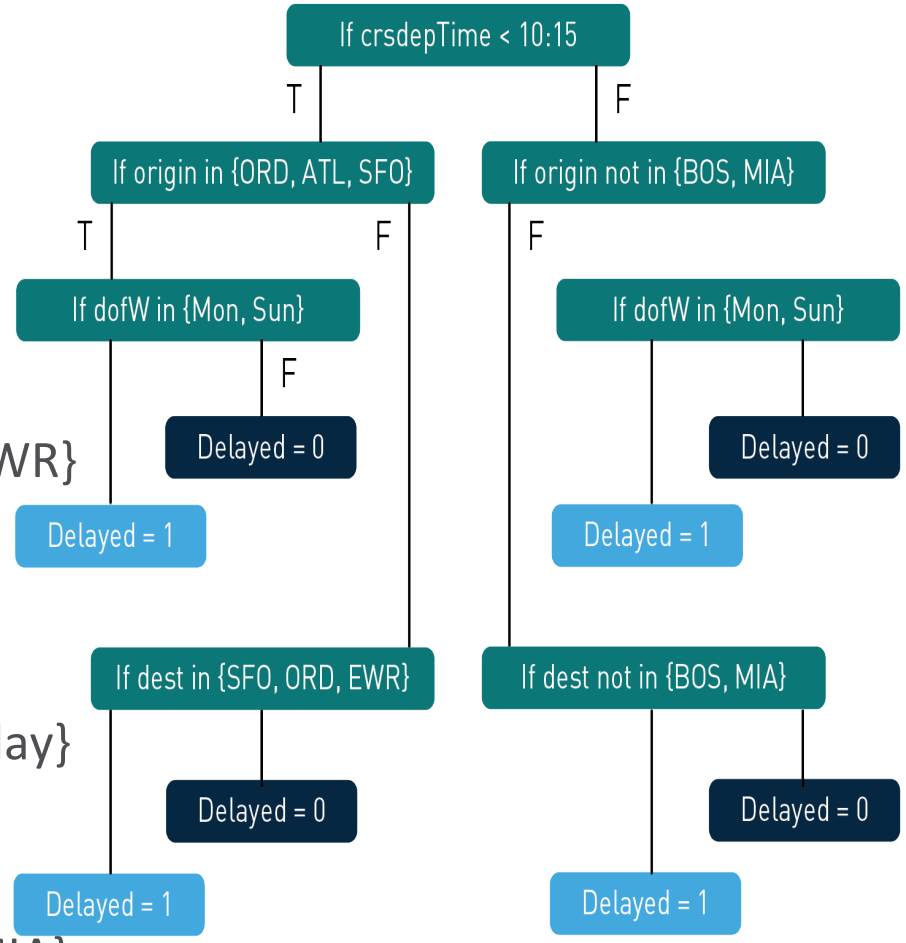
Decision Tree for Classification

- Tree of decisions about features
- IF THEN ELSE questions using features at each tree node
- Answers branch to child nodes

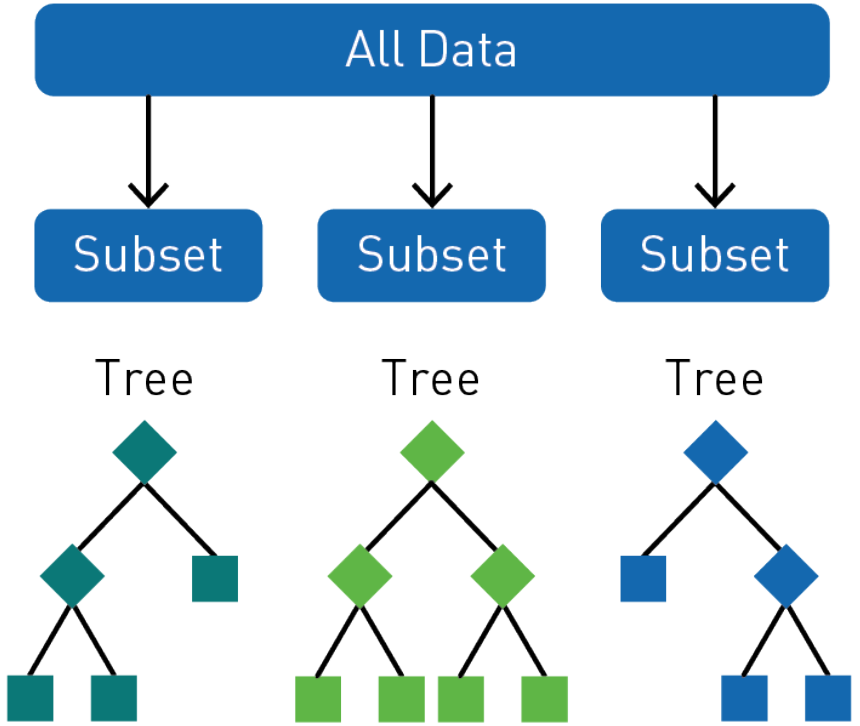


Decision Tree

- Q1: If the scheduled departure time < 10:15 AM
  - T:Q2: If the originating airport is in the set {ORD, ATL,SFO}
    - T:Q3: If the day of the week is in {Monday, Sunday}
      - T:Delayed=1
      - F: Delayed=0
    - F: Q3: If the destination airport is in the set {SFO,ORD,EWR}
      - T: Delayed=1
      - F: Delayed=0
  - F :Q2: If the originating airport is not in the set {BOS . MIA}
    - T:Q3: If the day of the week is in the set {Monday , Sunday}
      - T: Delayed=1
      - F: Delayed=0
    - F: Q3: If the destination airport is not in the set {BOS . MIA}
      - T: Delayed=1
      - F: Delayed=0



Random Forest



Featurization, Training and Model Selection

Feature Vectors are vectors of numbers representing the value for each feature

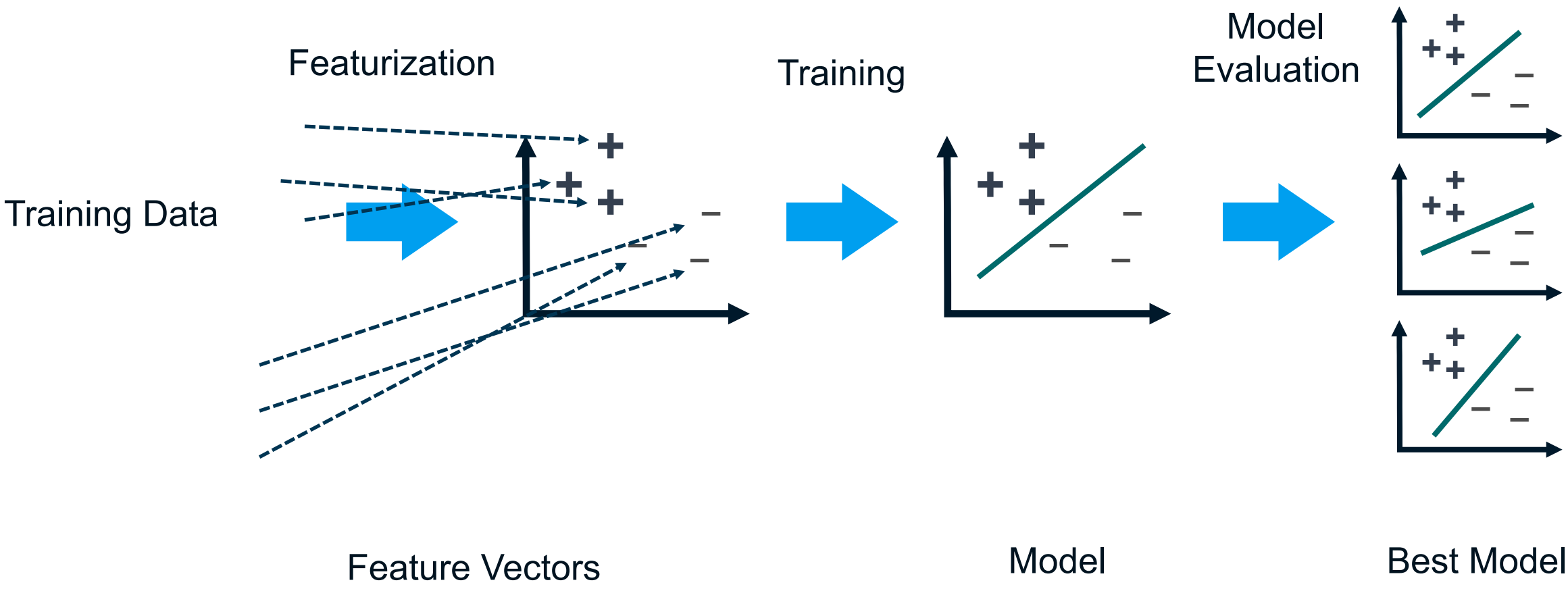
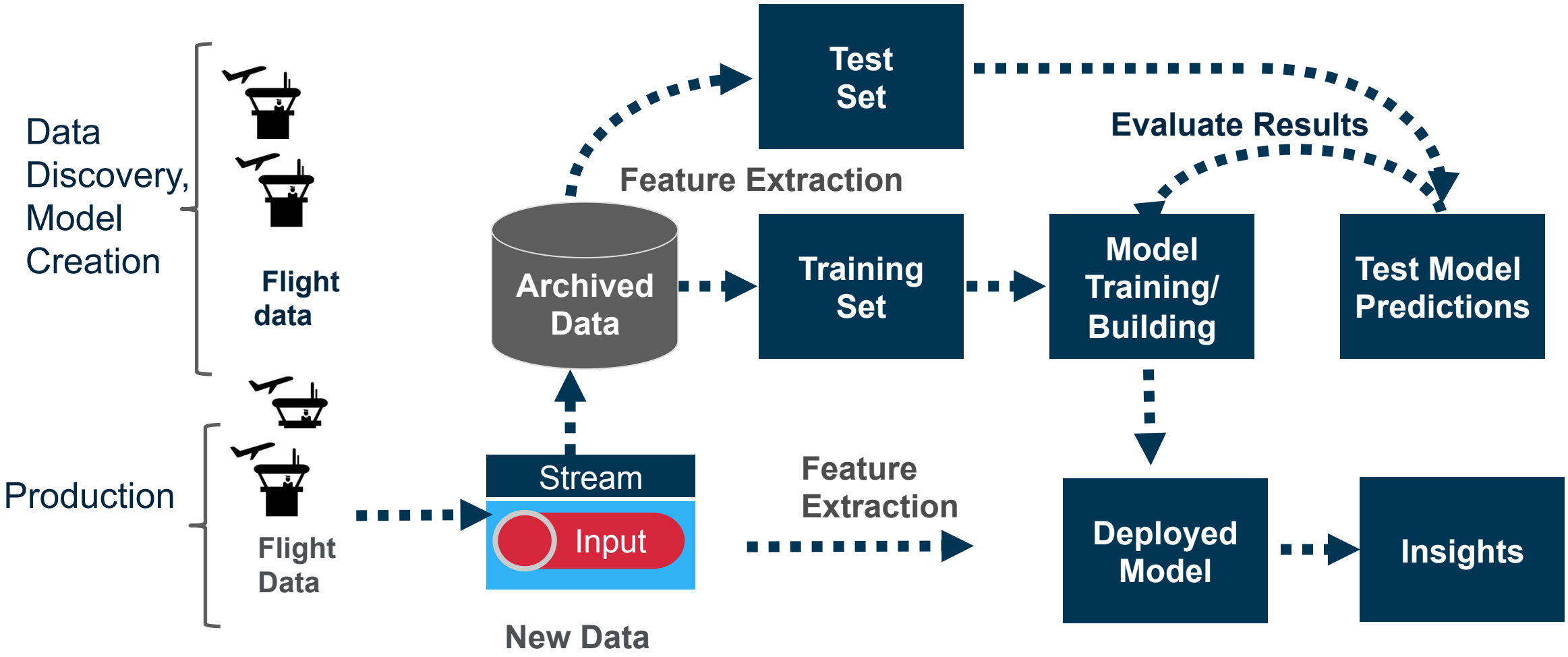
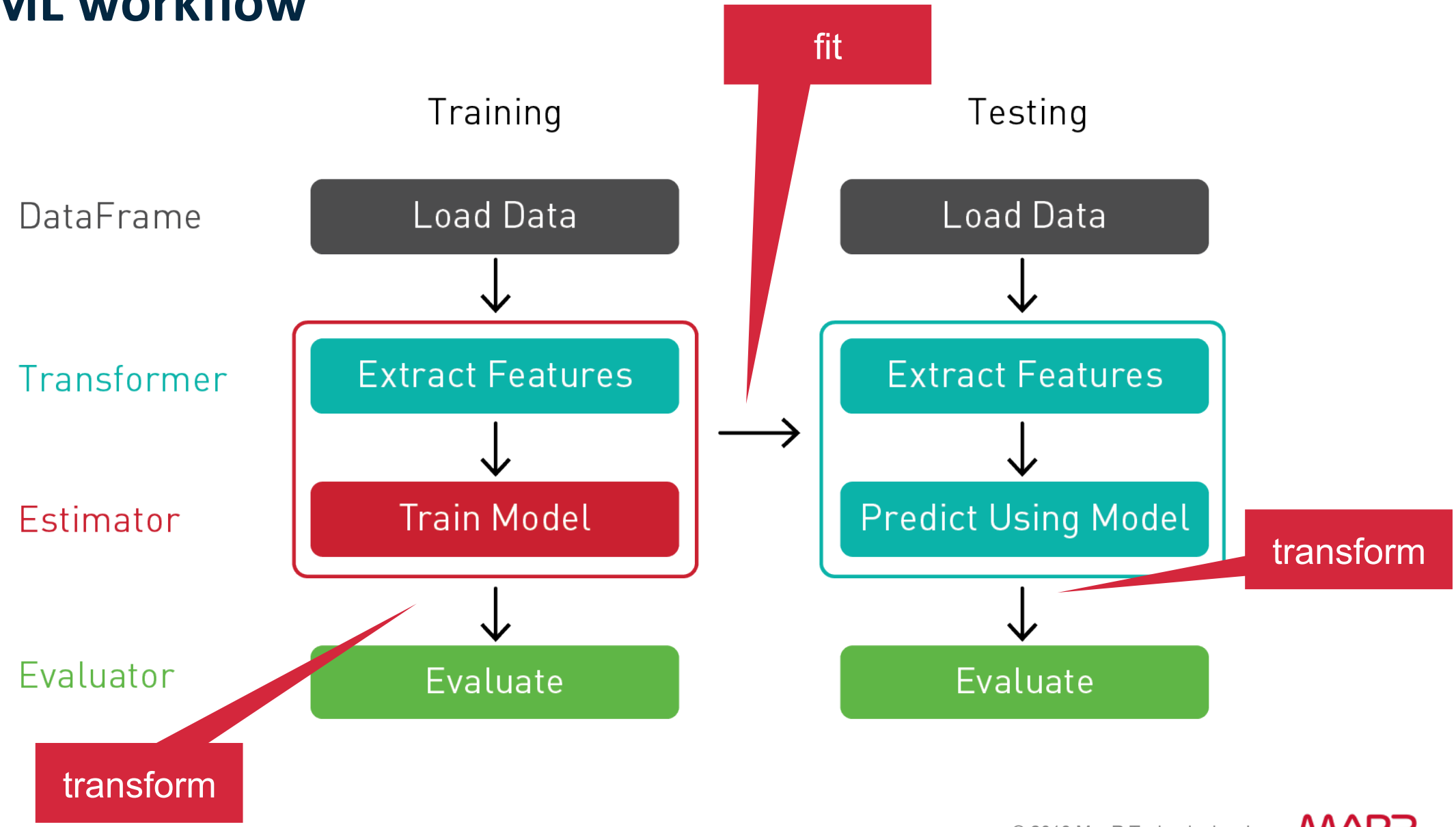


Image reference O'Reilly Learning Spark

ML Discovery Model Building

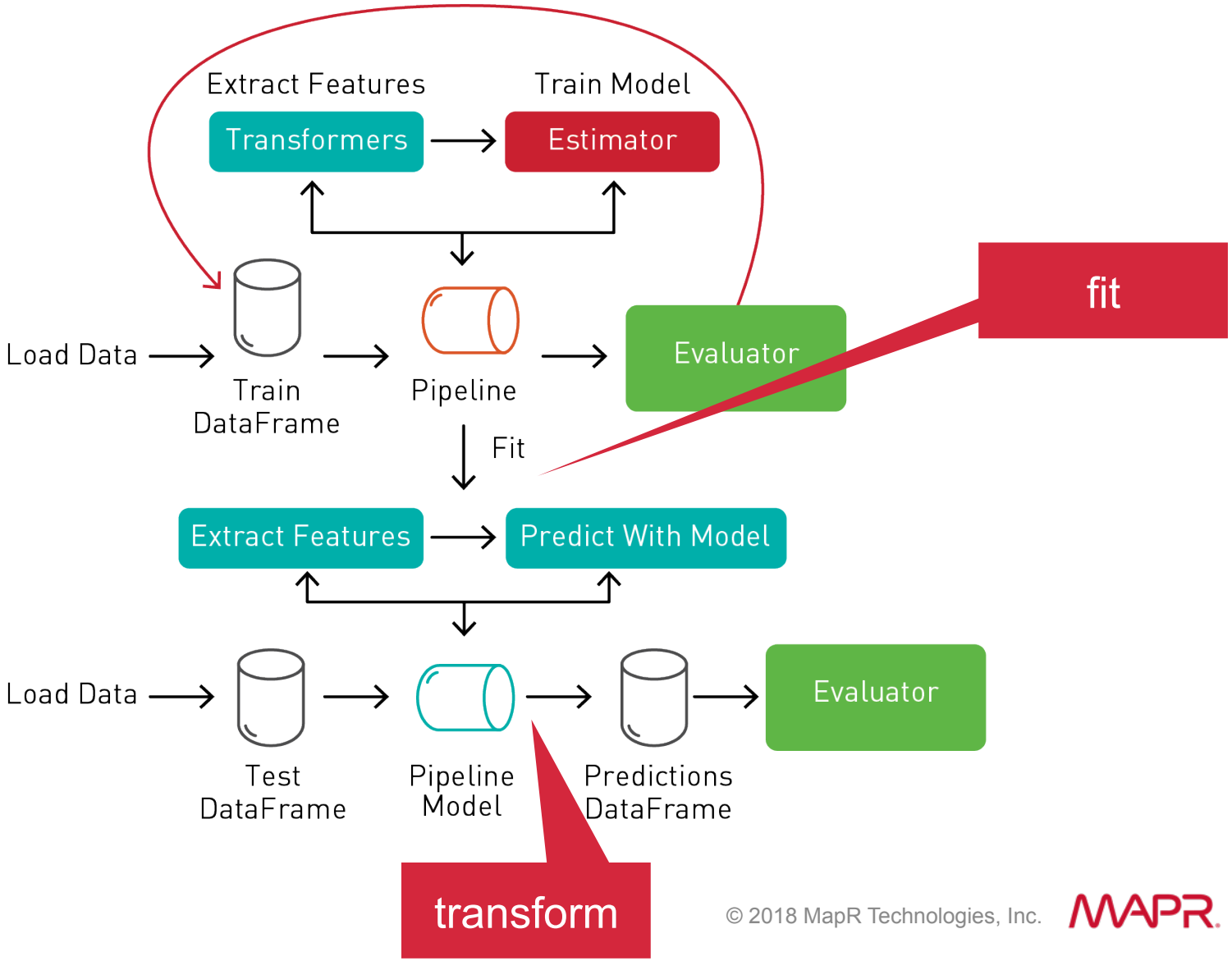


Spark ML workflow



Spark ML workflow with a Pipeline

A pipeline estimator fit returns a pipeline model



Extract the Features

Feature Vectors are vectors of numbers representing the value for each feature

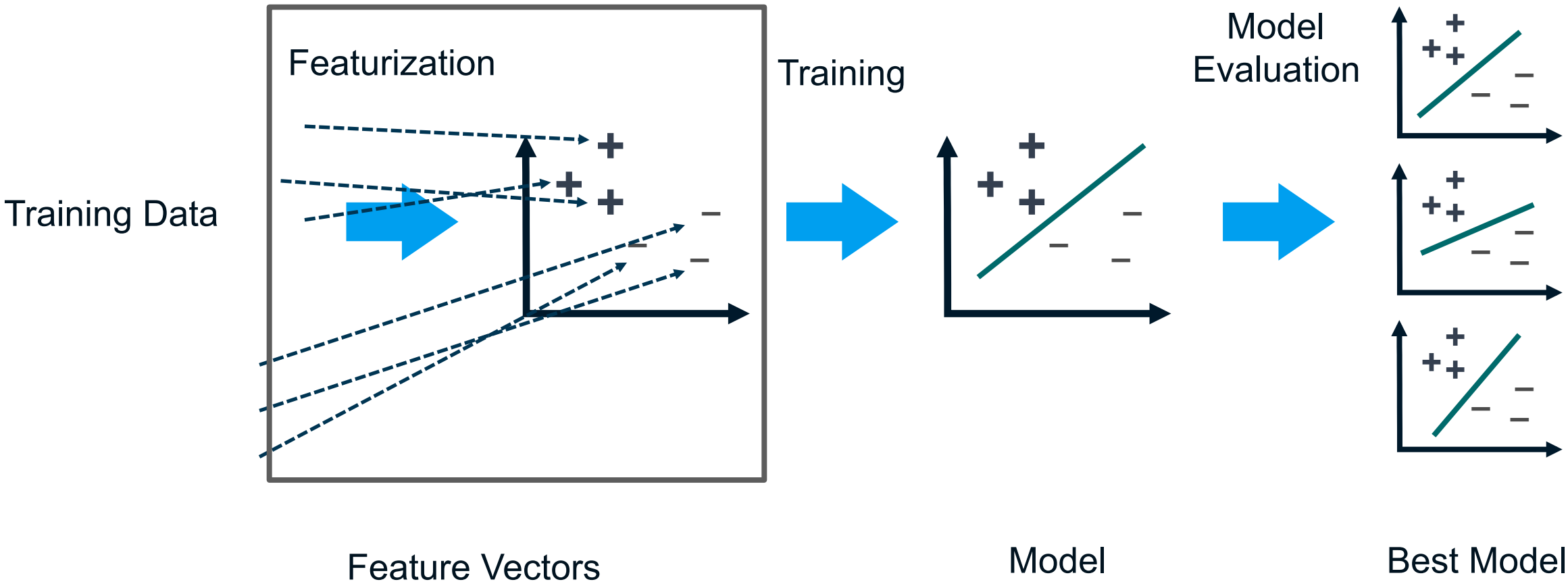
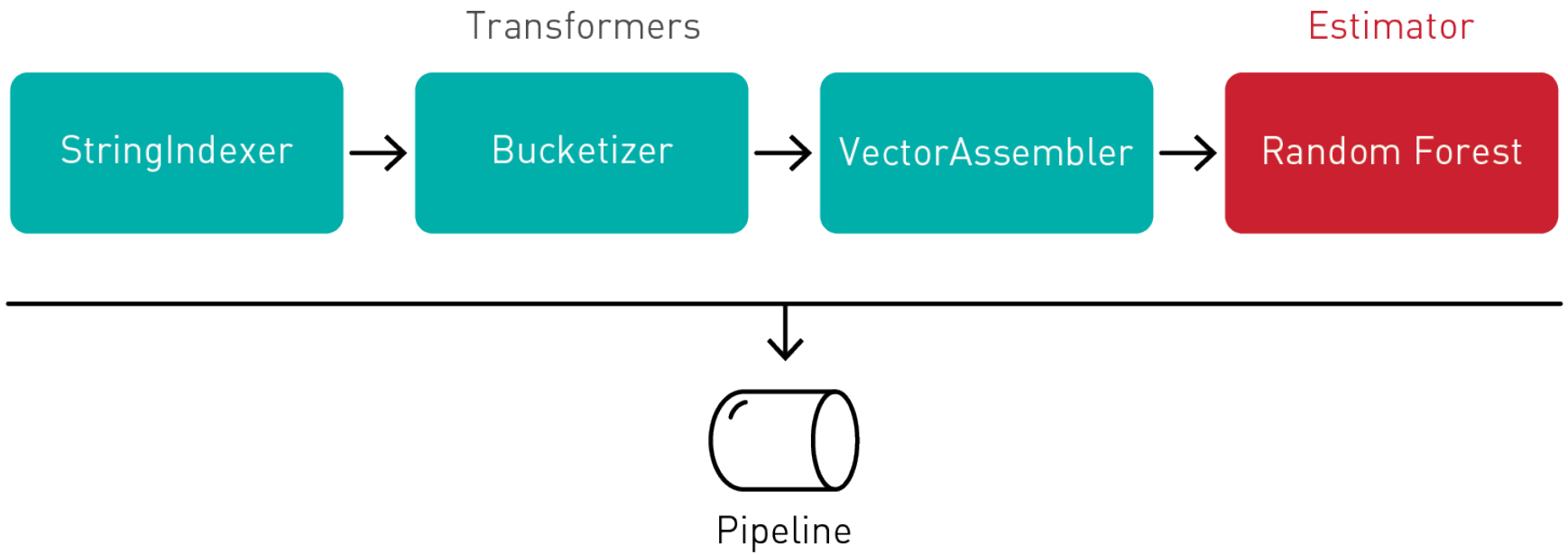


Image reference O'Reilly Learning Spark

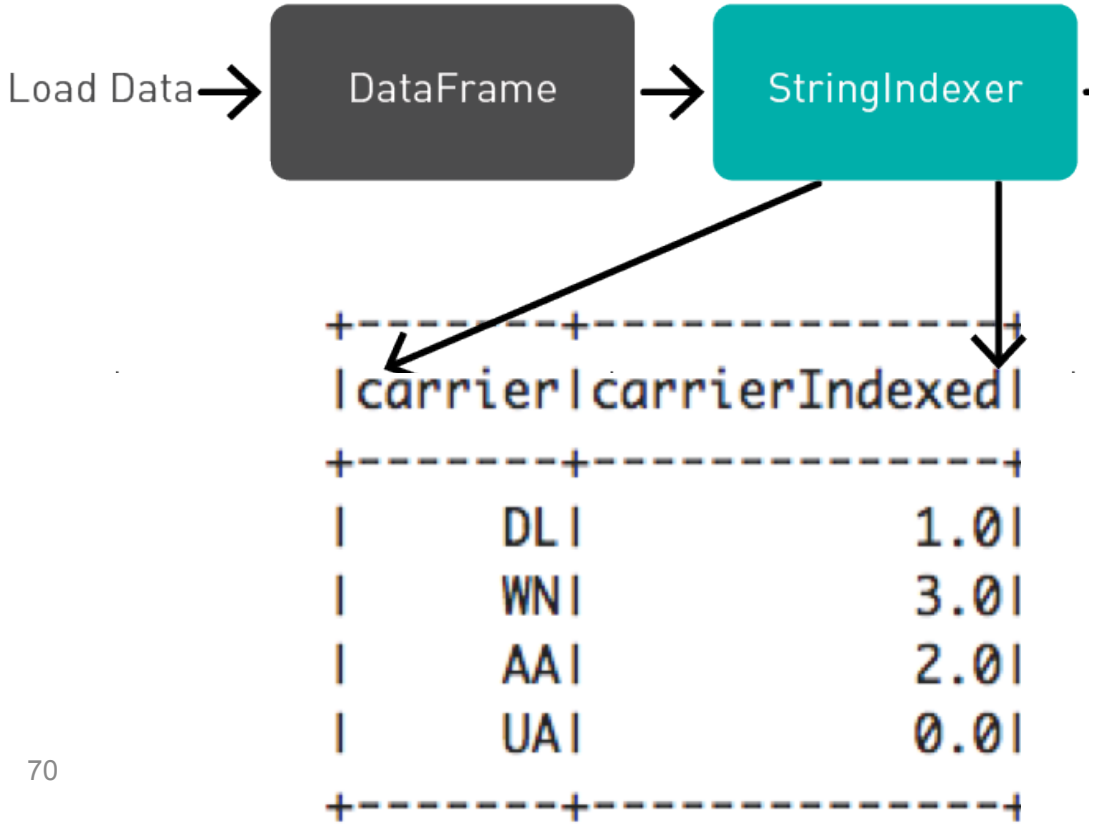
Set up Transformer Estimator Pipeline



A Pipeline chains multiple Transformers and Estimators together in a ML workflow

StringIndexer Transformer maps Strings to Numbers

```
val indexer = new StringIndexer()
      .setInputCol("carrier")
      .setOutputCol("carrierIndex")
Val df2=indexer.transform(df)
```



Transformer:

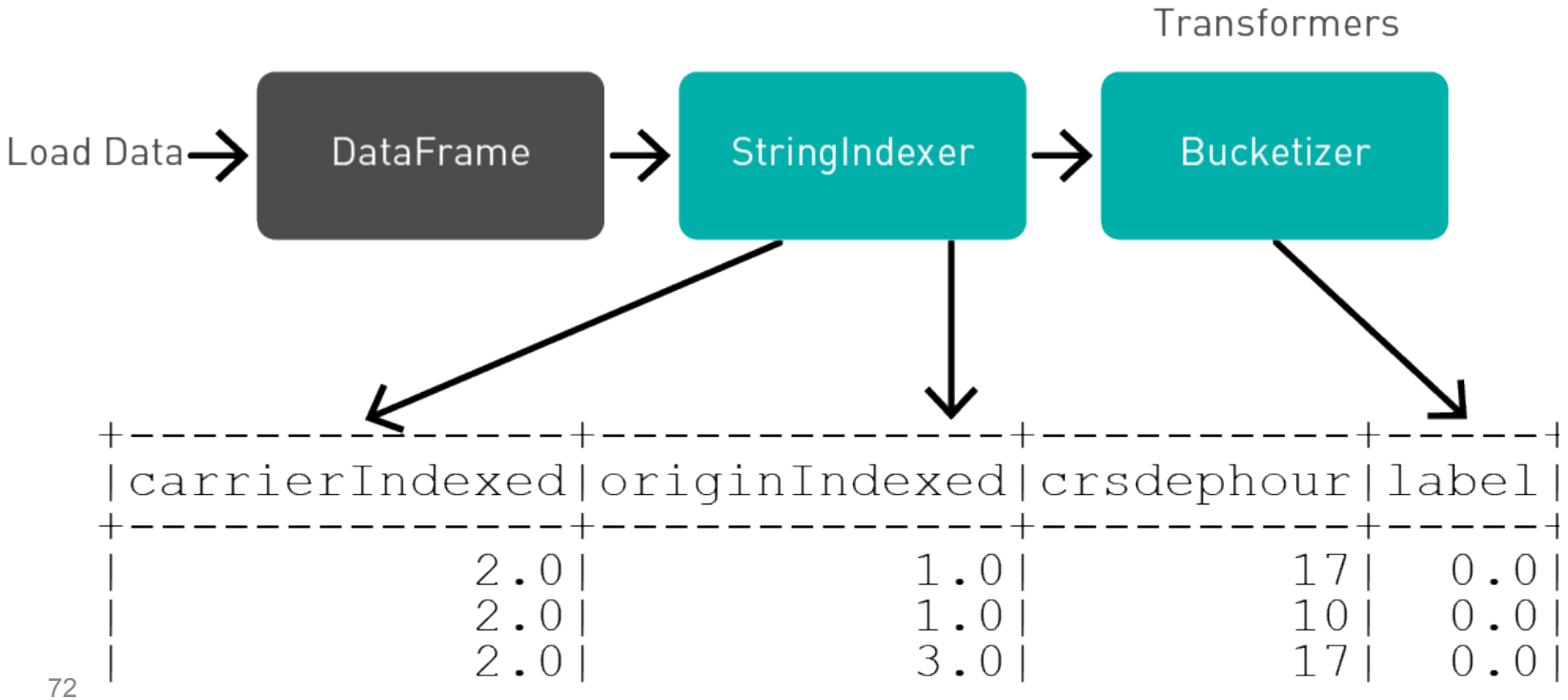
- **transform() method converts one DataFrame into another, by appending columns**

Create StringIndexers for all Categorical features

```
// column names for categorical feature columns
val categoricalColumns = Array("carrier", "origin", "dest",
"dofW", "orig_dest")

// create stringindexers for all string columns
val stringIndexers = categoricalColumns.map{ colName =>
      new StringIndexer()
        .setInputCol(colName)
        .setOutputCol(colName + "Indexed")
        .fit(strain)
}
```

Use Bucketizer Transformer to add a label 0 or 1

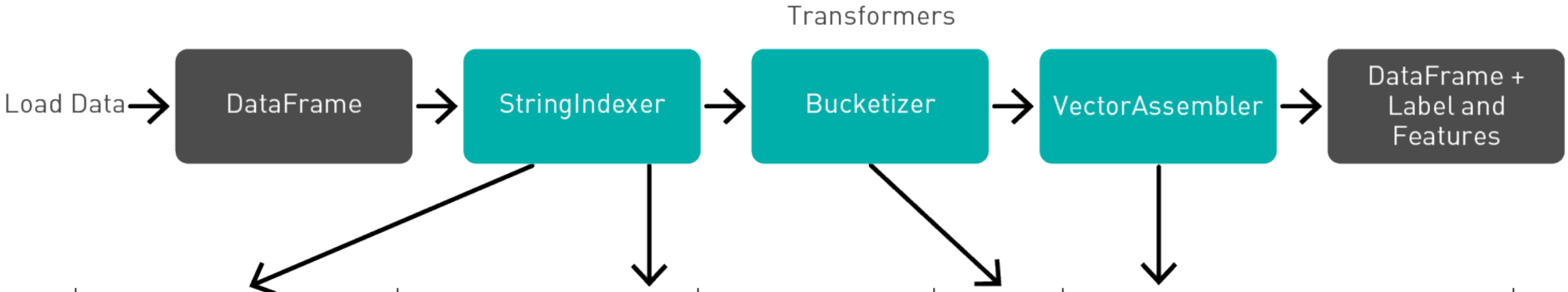
```
// add a label column based on departure delay
 val labeler = new Bucketizer().setInputCol("depdelay")
      .setOutputCol("label")
      .setSplits(Array(0.0,  40.0,
       Double.PositiveInfinity)
```




```
+---------------+--------------+-----------+-----+
|carrierIndexed|originIndexed|crsdephour|label|
+---------------+--------------+-----------+-----+
|            2.0|           1.0|         17|  0.0|
|            2.0|           1.0|         10|  0.0|
|            2.0|           3.0|         17|  0.0|
```

Use Vector Assembler to add features

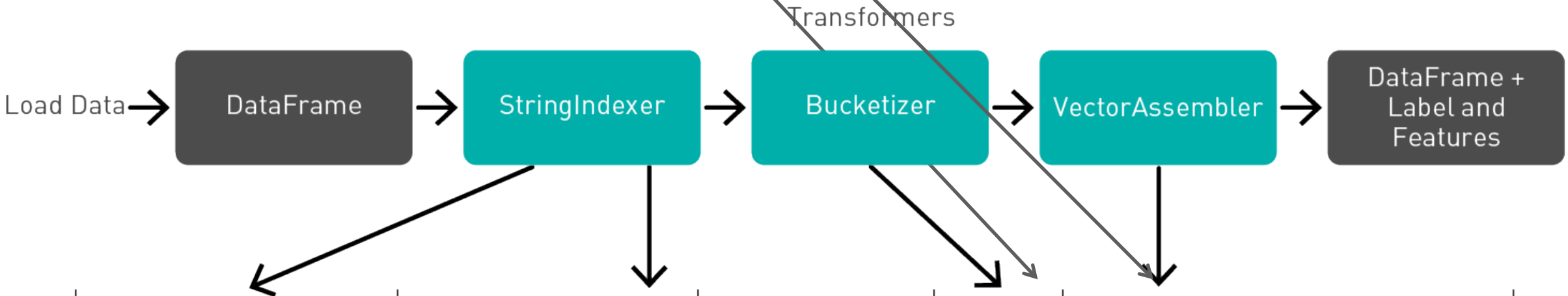
```
 val featureCols = Array("carrierIndexed", "destIndexed”,"originIndexed",
"dofWIndexed", "orig_destIndexed”,"crsdephour", "crsdeptime”,"crsarrtime”,
"crselapsedtime", "dist")
// combines a list of feature columns into a vector column
 val assembler = new VectorAssembler()
      .setInputCols(featureCols)
      .setOutputCol("features")
```




```
+---------------+--------------+----------+-----+--------------------+
|carrierIndexed|originIndexed|crsdephour|label|features            |
+--------------+--------------+----------+-----+--------------------+
|           2.0|           1.0|        17|  0.0|[2.0,3.0,1.0,2.0,...|
|           2.0|           1.0|        10|  0.0|[2.0,0.0,1.0,2.0,...|
```

Create RandomForest Estimator to train our model

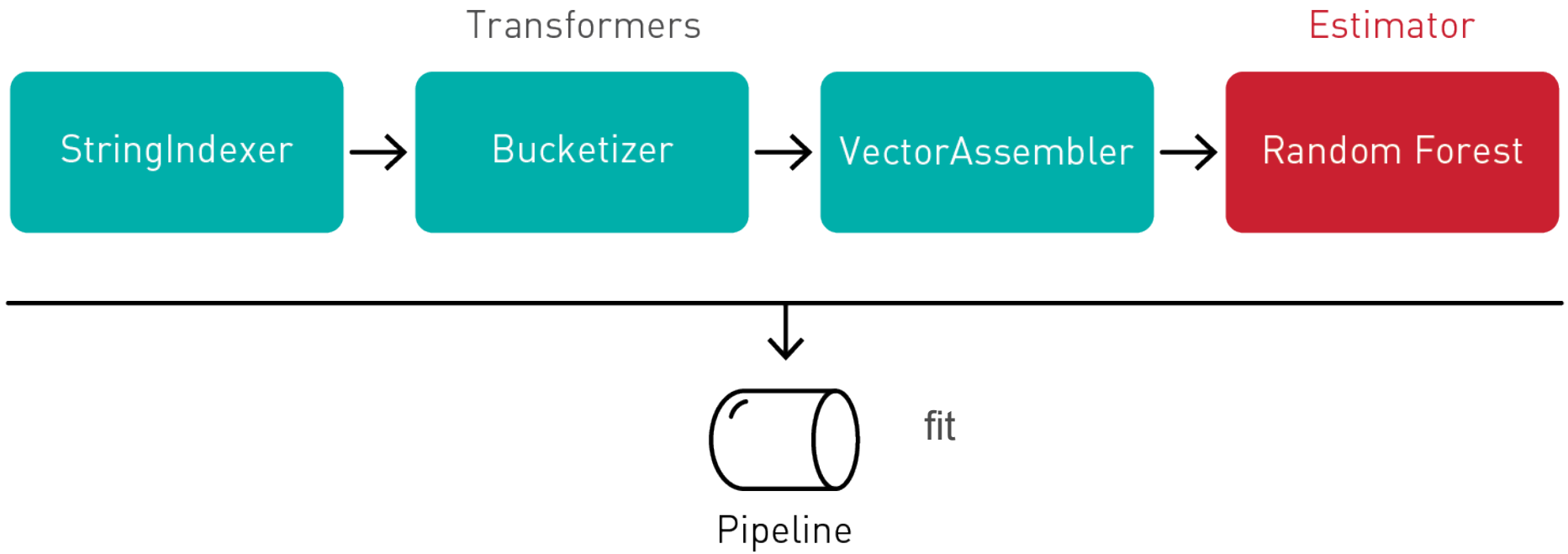
```
val rf = new RandomForestClassifier()
       .setLabelCol("label")
       .setFeaturesCol("features")
```




```
+---------------+--------------+----------+-----+--------------------+
|carrierIndexed|originIndexed|crsdephour|label|features            |
+--------------+--------------+----------+-----+--------------------+
|           2.0|           1.0|        17|  0.0|[2.0,3.0,1.0,2.0,...|
|           2.0|           1.0|        10|  0.0|[2.0,0.0,1.0,2.0,...|
|           2.0|           3.0|        17|  0.0|[2.0,0.0,3.0,2.0,...|
```

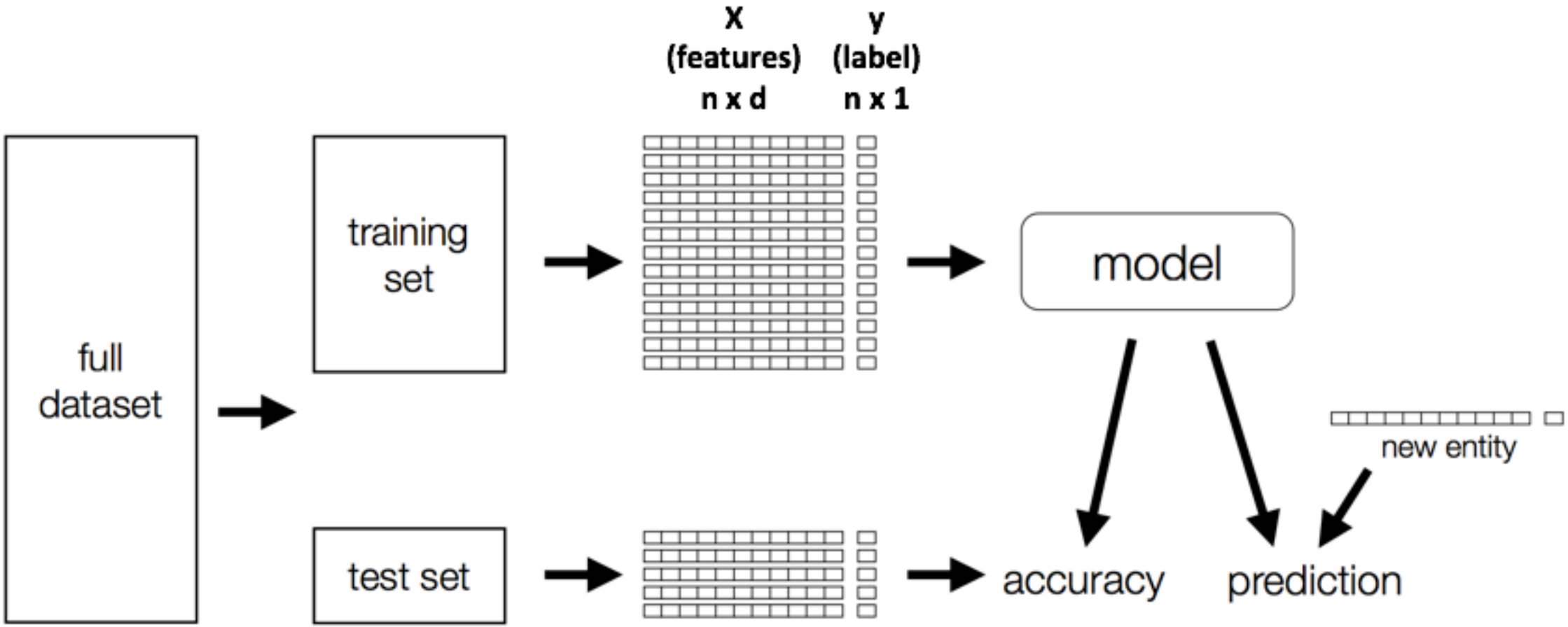
Put Feature Transformers and Estimator in Pipeline

```
val steps = stringIndexers  ++ Array(labeler, assembler, rf)

val pipeline = new Pipeline().setStages(steps)
```



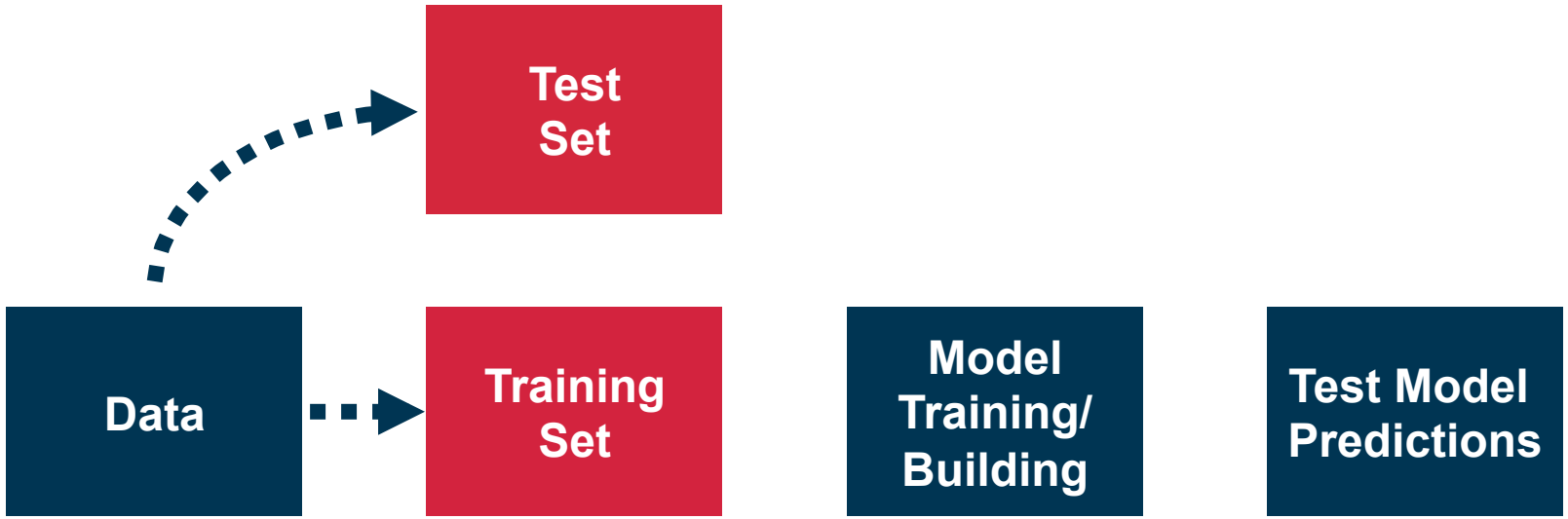
Estimator pipeline fit() method trains on a dataset and produces a model

Evaluate the Predictions from the model



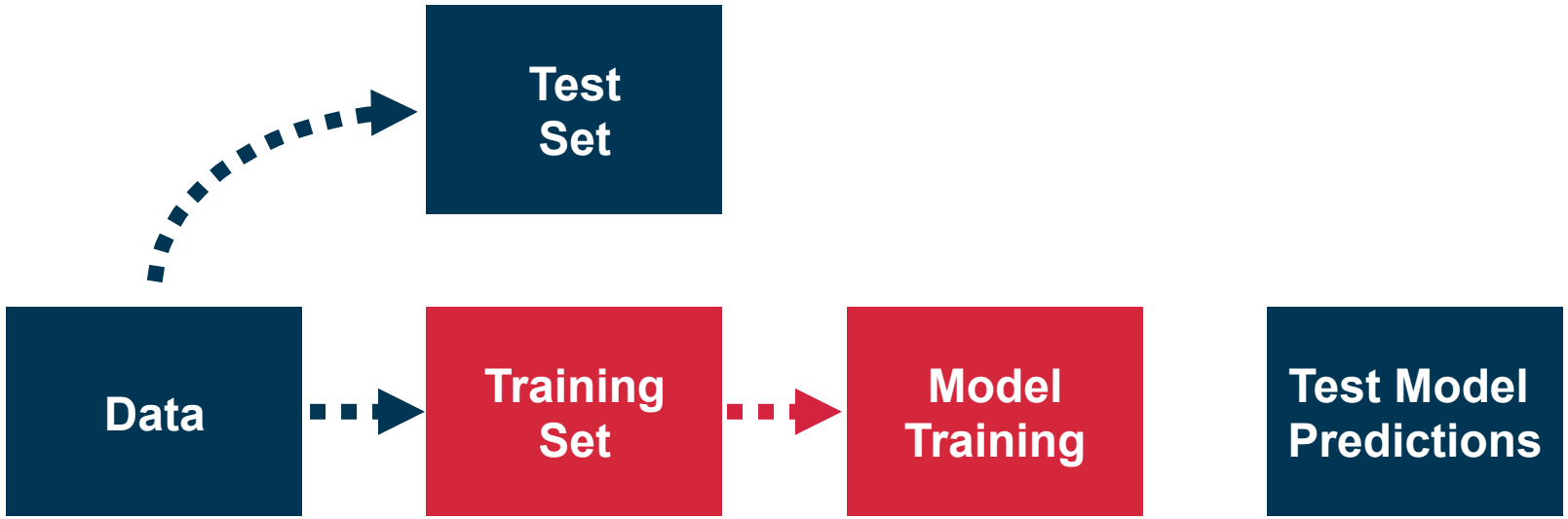
K-fold Cross-Validation Process

data is randomly split into K partition training and test dataset pairs



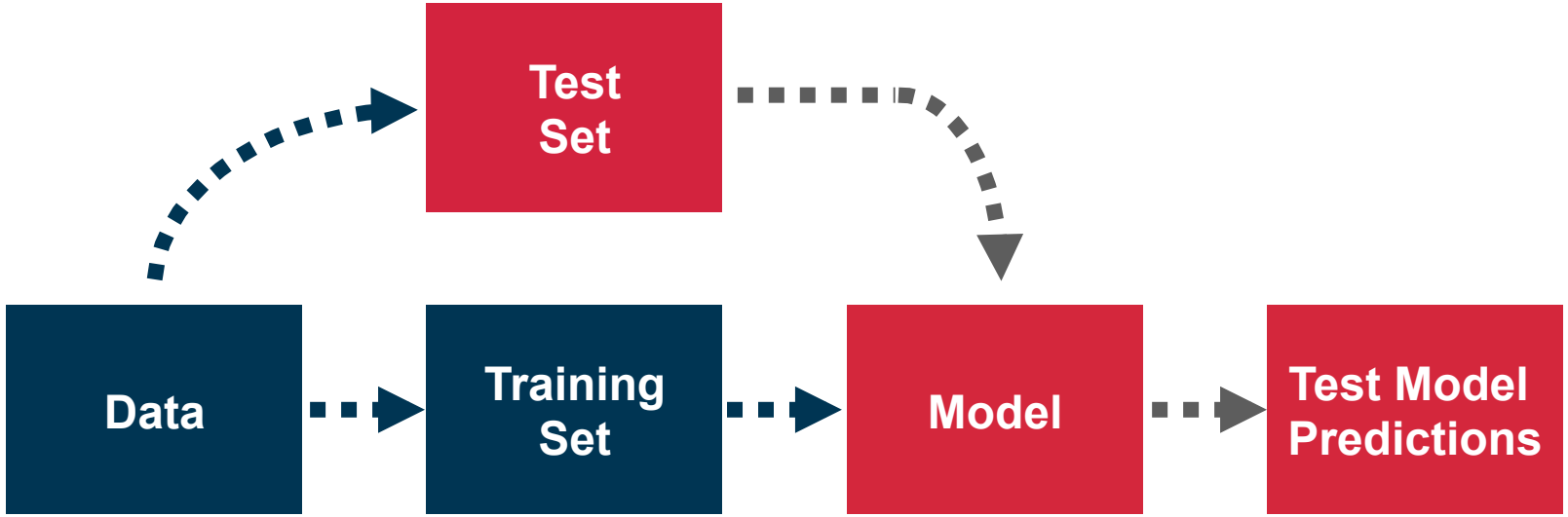
K-fold Cross-Validation Process

Train algorithm with training dataset



ML Cross-Validation Process

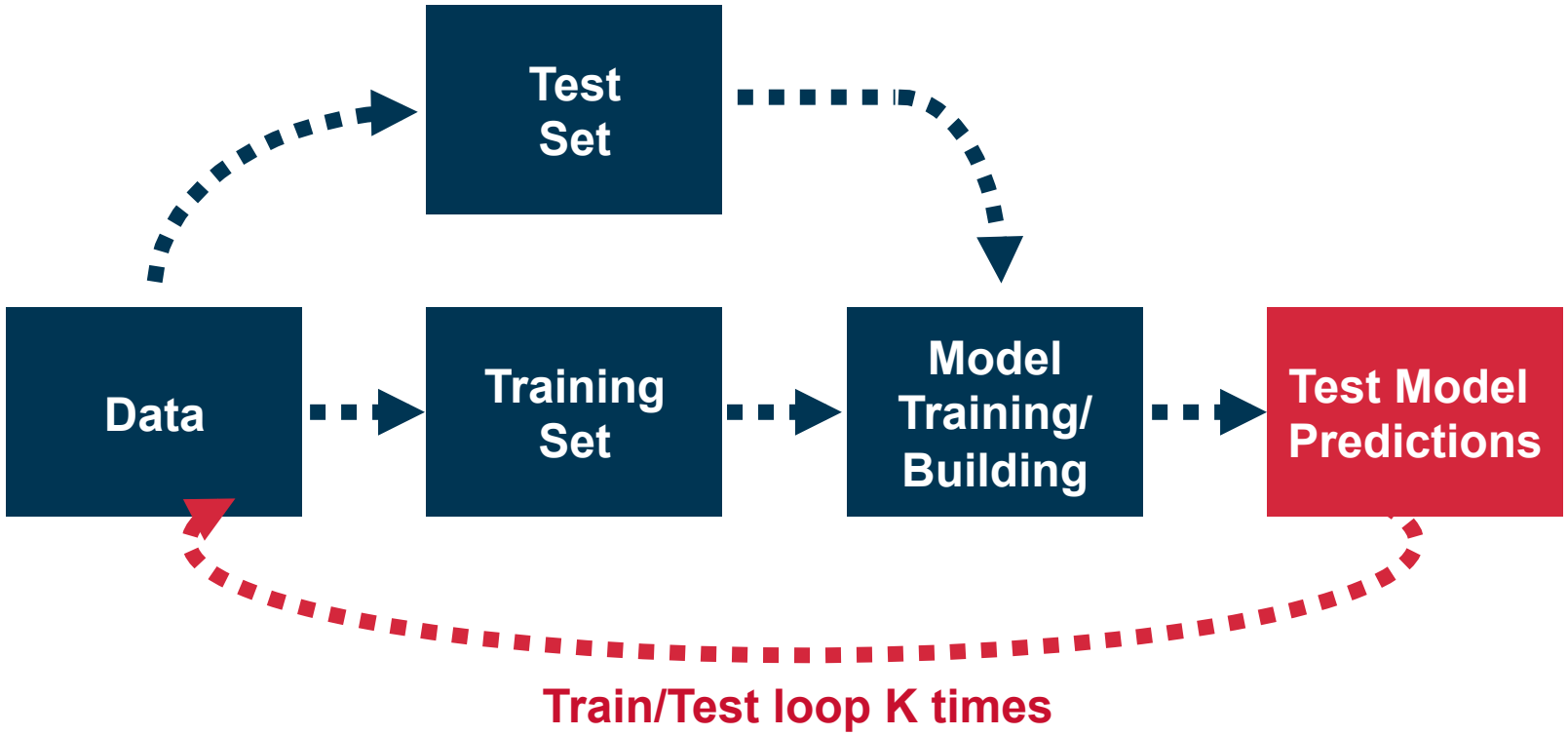
Evaluate the model with the Test Set



K-fold Cross-Validation Process

Repeat K times

select the Model produced by the best-performing set of parameters

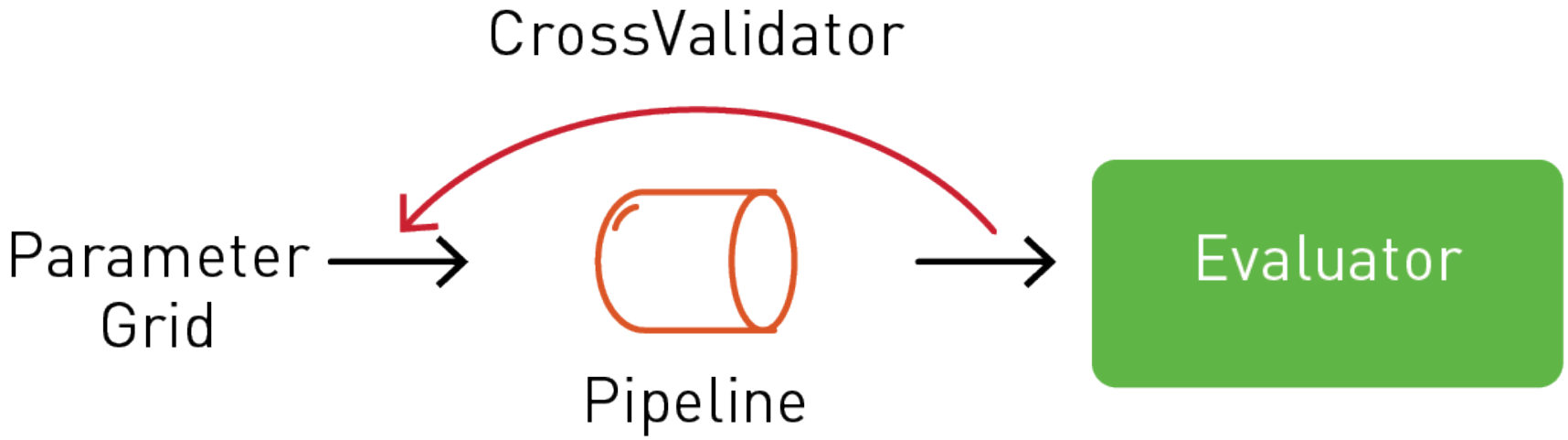


Cross Validation transformation estimation pipeline

Set up a CrossValidator with:

- **Parameter grid**, Estimator (pipeline), Evaluator

Perform grid search based model selection



Parameter Tuning with CrossValidator with a Paramgrid

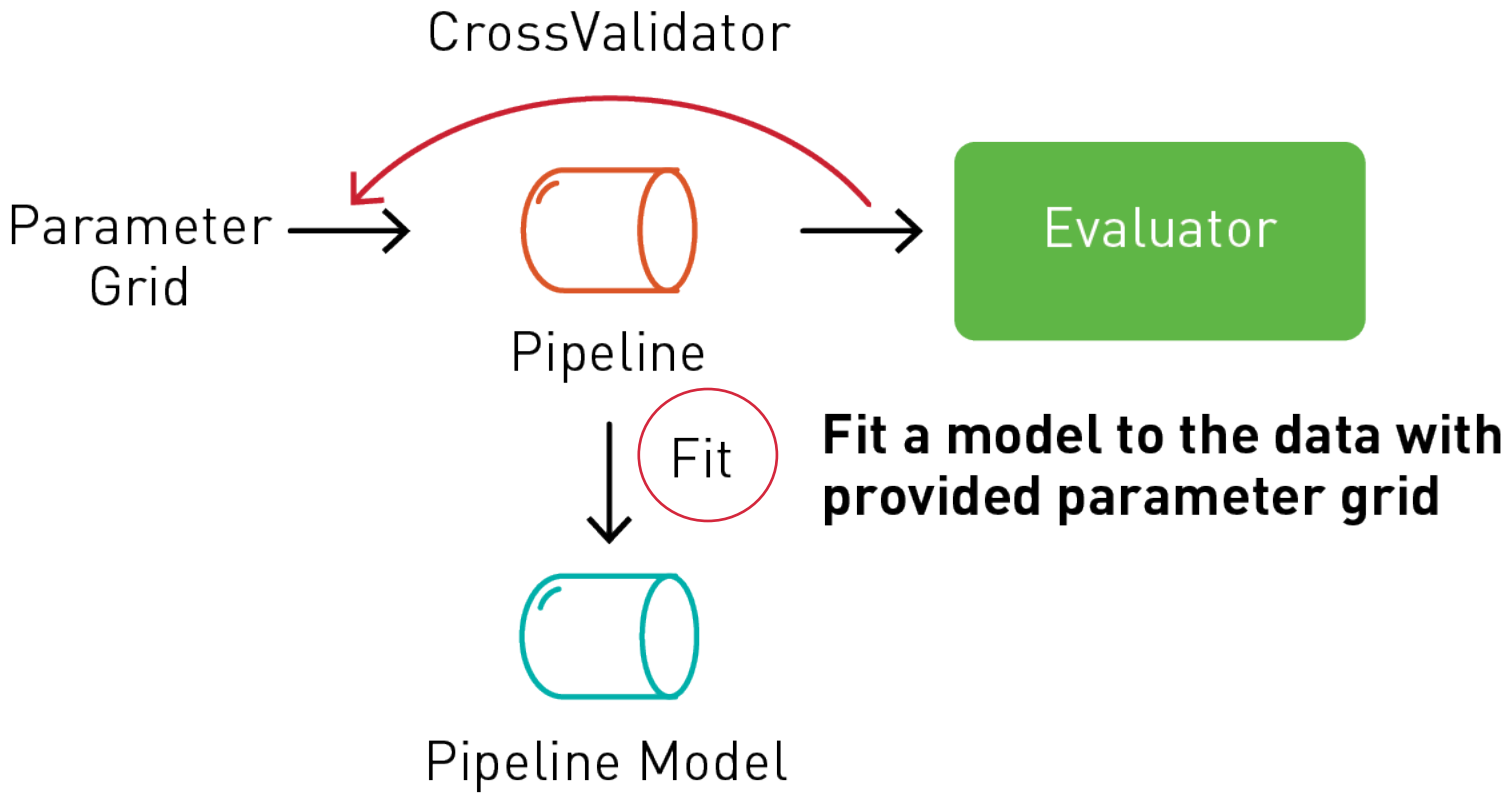
CrossValidator

- Given:
  - Estimator
  - **Parameter grid**
  - Evaluator
- Find best parameters and model

```
val paramGrid = new ParamGridBuilder()
    .addGrid(rf.maxBins, Array(100, 200))
    .addGrid(rf.maxDepth, Array(2, 4, 10))
    .addGrid(rf.numTrees, Array(5, 20))
    .addGrid(rf.impurity, Array("entropy", "gini"))
    .build()

val evaluator= new BinaryClassificationEvaluator()
      .setLabelCol("label")
      .setRawPredictionCol("prediction")

val crossval = new CrossValidator()
      .setEstimator(pipeline)
      .setEvaluator(evaluator)
      .setEstimatorParamMaps(paramGrid)
      .setNumFolds(3)
```

Crossvalidator fit a Model

```
val pipelineModel = crossvalidator.fit(train)
```



Random Forest Feature Importances

```
val featureImportances = pipelineModel.bestModel.asInstanceOf[PipelineModel]
  .stages(stringIndexers.size + 2)
  .asInstanceOf[RandomForestClassificationModel].featureImportances

assembler.getInputCols.zip(featureImportances.toArray).sortBy(-_._2).foreach { case (feat, imp) =>
println(s"feature: $feat, importance: $imp") }
```

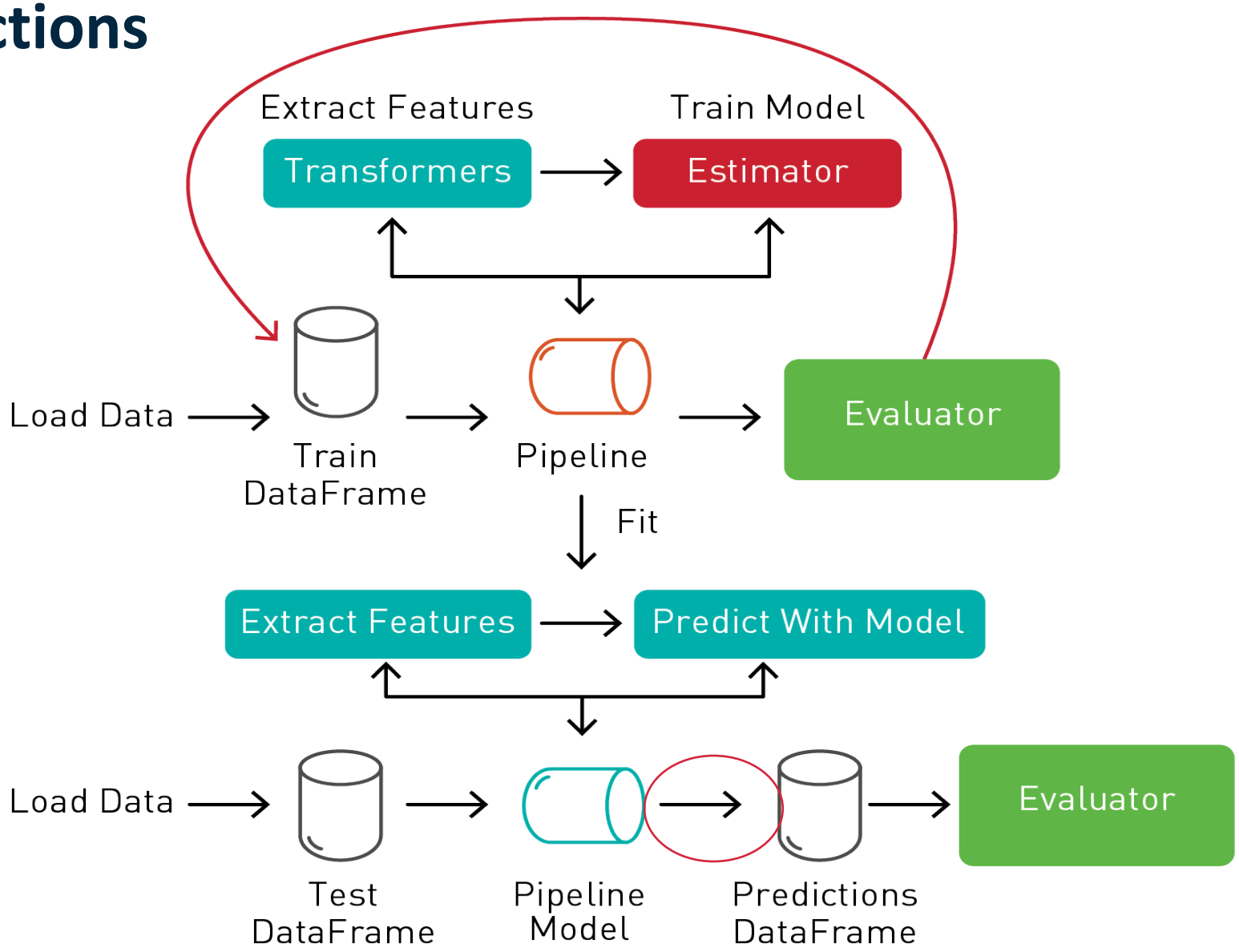
result:

feature: crsdeptime, importance: 0.295432101974887 4

feature: orig_destIndexed, importance: 0.2154067691316247 6

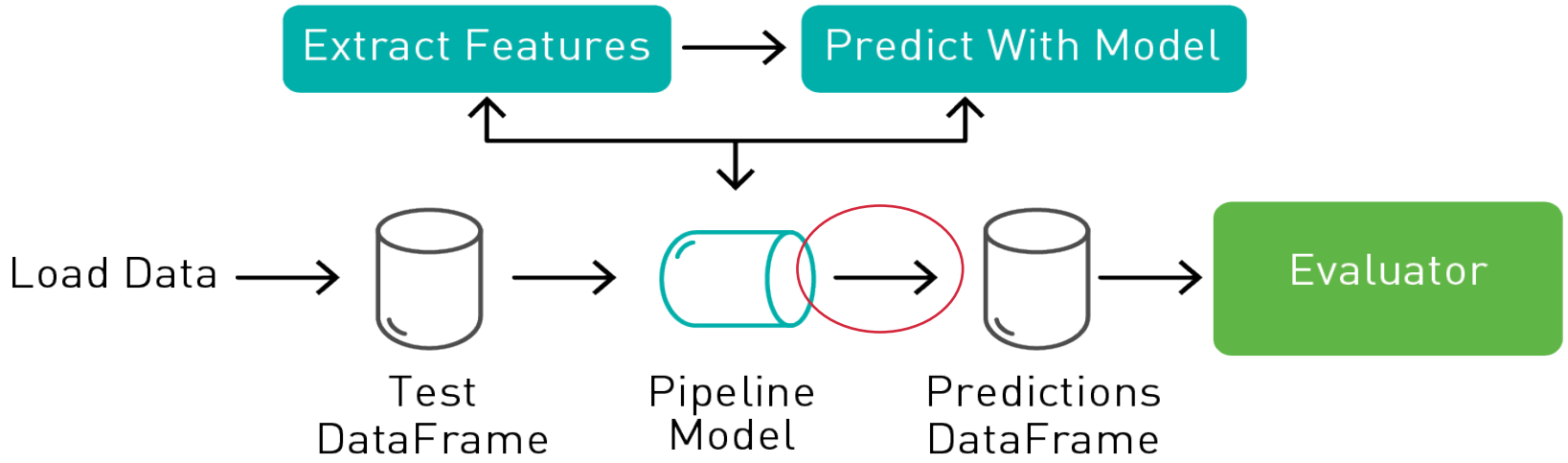
feature: crsarrtime, importance: 0.1594826730807351
feature: crsdephour, importance: 0.1123275083502450 8
feature: destIndexed, importance: 0.0706885195251565 8
feature: carrierIndexed, importance: 0.0373706756139363 5
feature: dist, importance: 0.036751142051444 13
feature: dofWIndexed, importance: 0.0301185279127827 44
feature: originIndexed, importance: 0.0224015212726978 23
feature: crselapsedtime, importance: 0.0200205610864901 13

Get Test Predictions



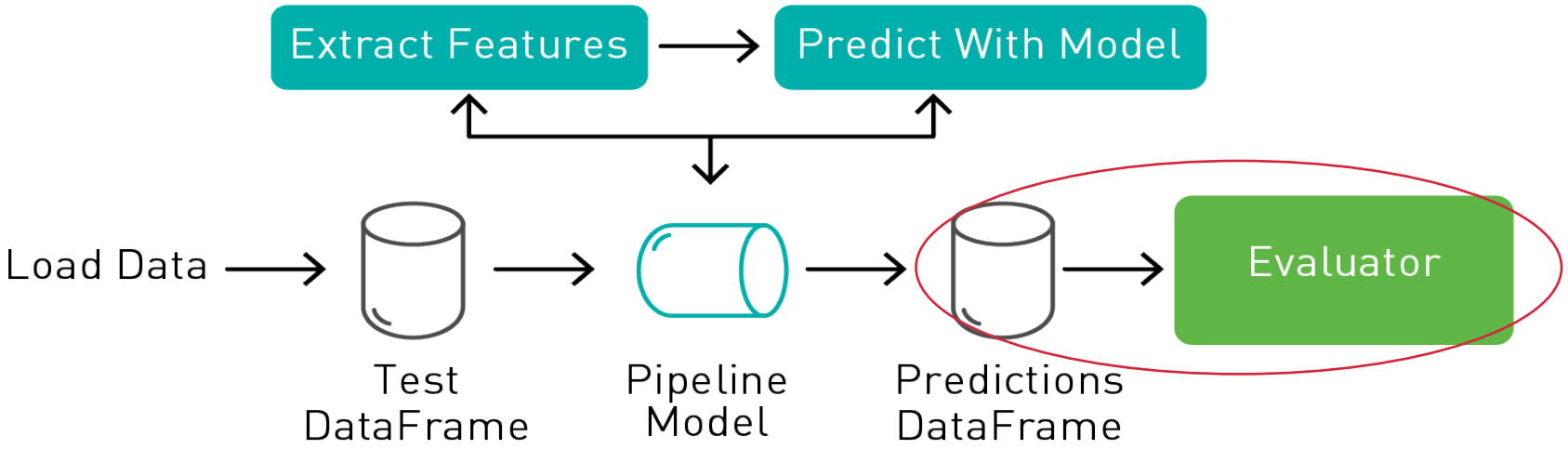
Get Predictions on Test Data

```
val predictions = pipelineModel.transform(test)
```



Evaluate the Predictions from the model

```
val areaUnderROC = evaluator.evaluate(predictions)
```



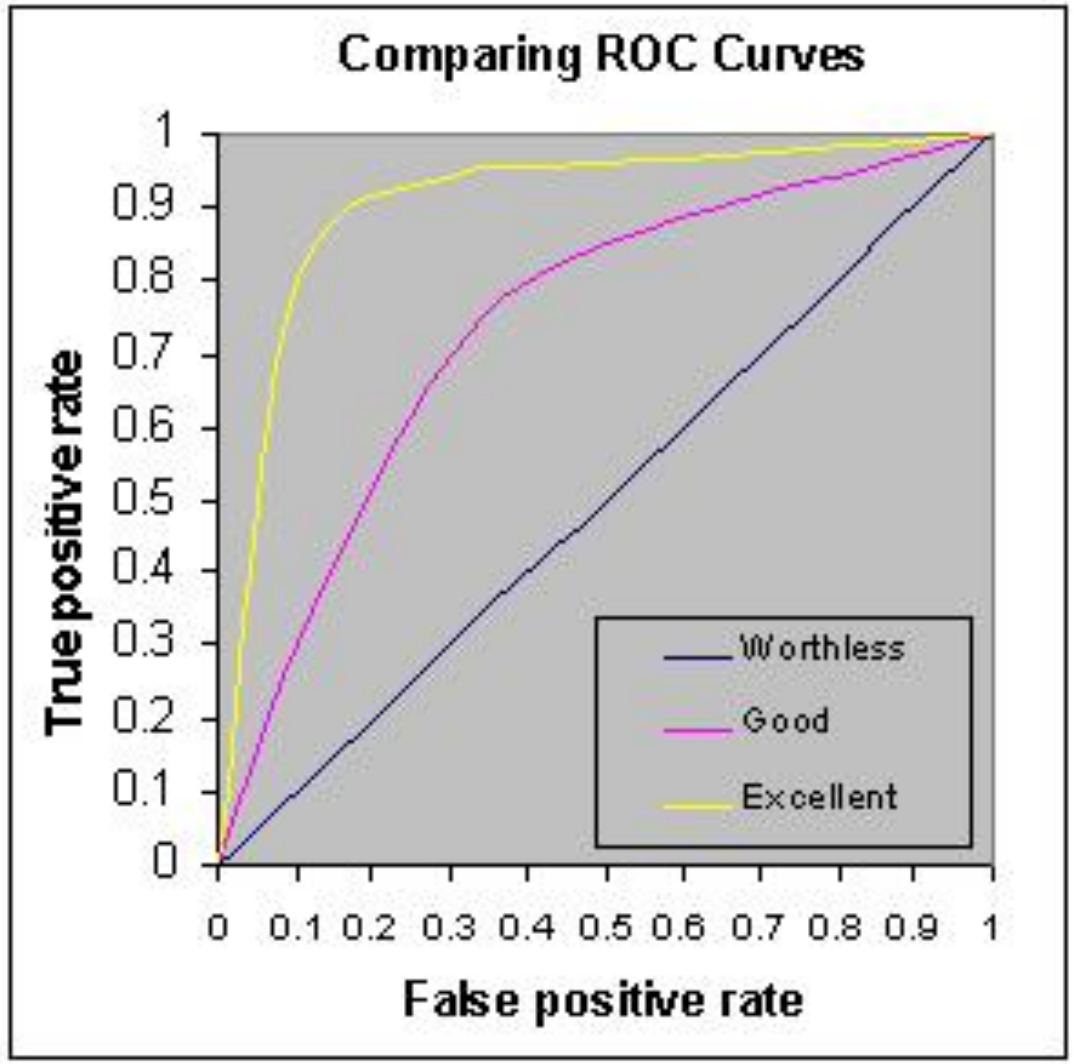

```
+-----+--------------------+----------+
|label|            features|prediction|
+-----+--------------------+----------+
|  1.0|[117.0,0.0,0.0,18...|       1.0|
|  0.0|[65.0,0.0,0.0,129...|       0.0|
```

Area under the ROC curve

Accuracy is measured by the area under the ROC curve.

The area measures correct classifications

- An area of 1 represents a perfect test
- an area of .5 represents a worthless test



Calculate Metrics

```
val lp = predictions.select( "label", "prediction")
val counttotal = predictions.count()
val correct = lp.filter($"label" === $"prediction").count()
val wrong = lp.filter(not($"label" === $"prediction")).count()
val truep = lp.filter($"prediction" === 0.0).filter($"label" === $"prediction").count() /
counttotal.toDouble
val truen = lp.filter($"prediction" === 1.0).filter($"label" === $"prediction").count()/
counttotal.toDouble
val falsen = lp.filter($"prediction" === 0.0).filter(not($"label" === $"prediction")).count()/
counttotal.toDouble
val falsep = lp.filter($"prediction" === 1.0).filter(not($"label" === $"prediction")).count()/
counttotal.toDouble
val ratioWrong=wrong.toDouble/counttotal.toDouble
val ratioCorrect=correct.toDouble/counttotal.toDouble
```

Calculate Metrics

ratio Correct0.6129679791041889

ratio wrong 0.3870320208958112

true positive 0.537994582567476

true negative 0.0749733965367128

false negative 0.03526168133879754

false positive 0.35177033955693143

Precision Grid

prediction/label	Actual Not Delayed	Actual Delayed
Predicted Not Delayed	.53 (TP)	.035 (FN)
Predicted Delayed	.35 (FP)	.074 (TN)

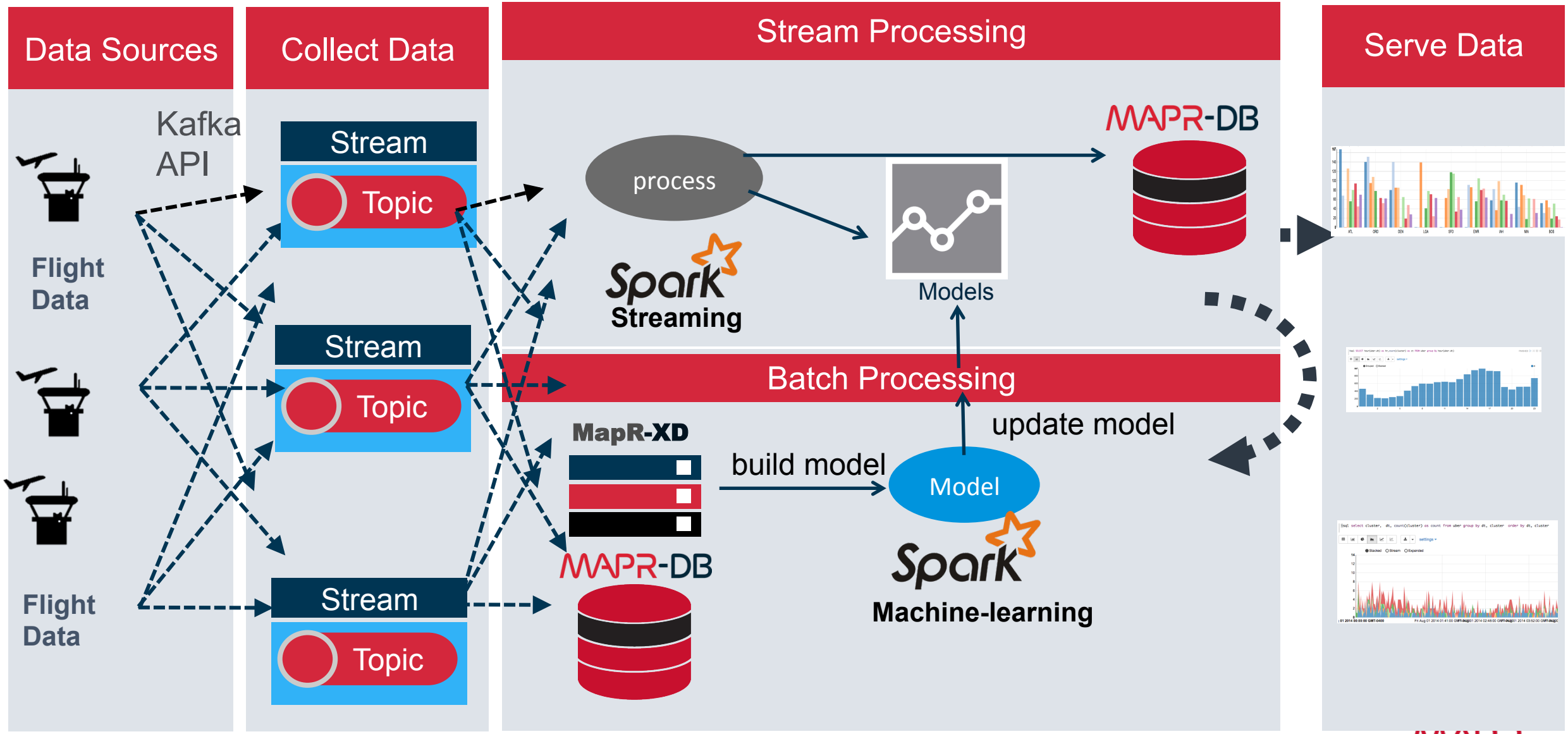
Save the Model

```
pipelineModel.write.overwrite().save(”/pathl")
```

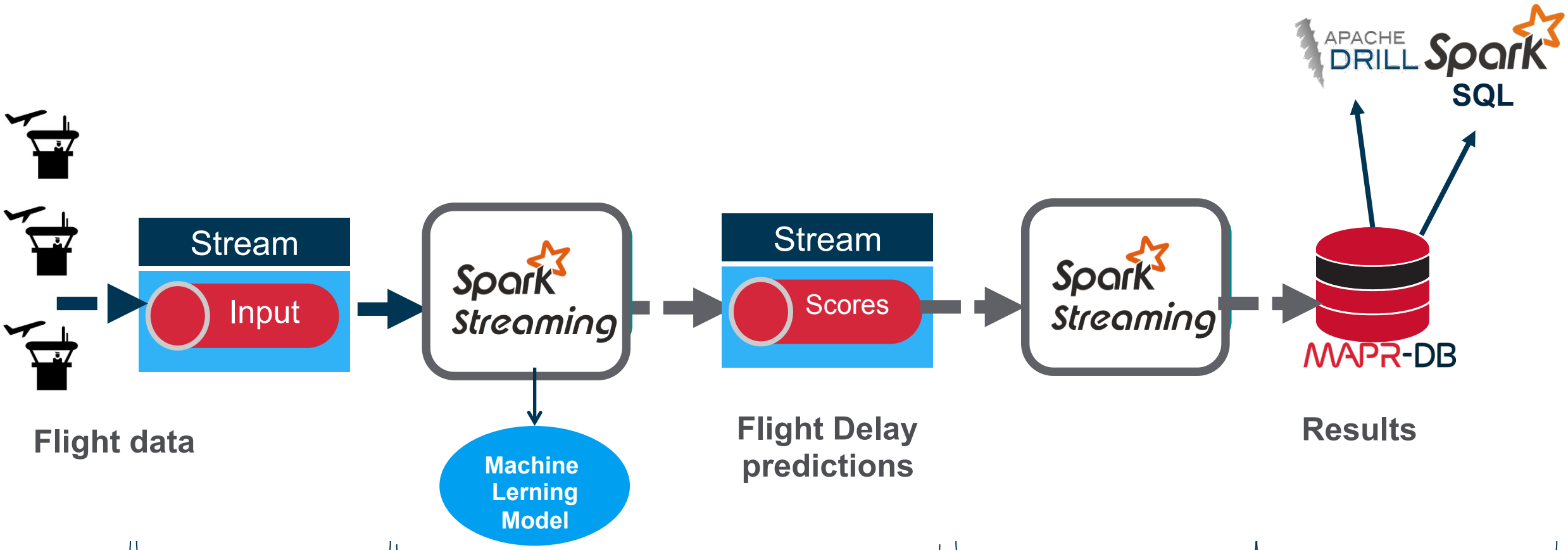
Later

```
val sameModel = CrossValidatorModel.load(“/path")
```

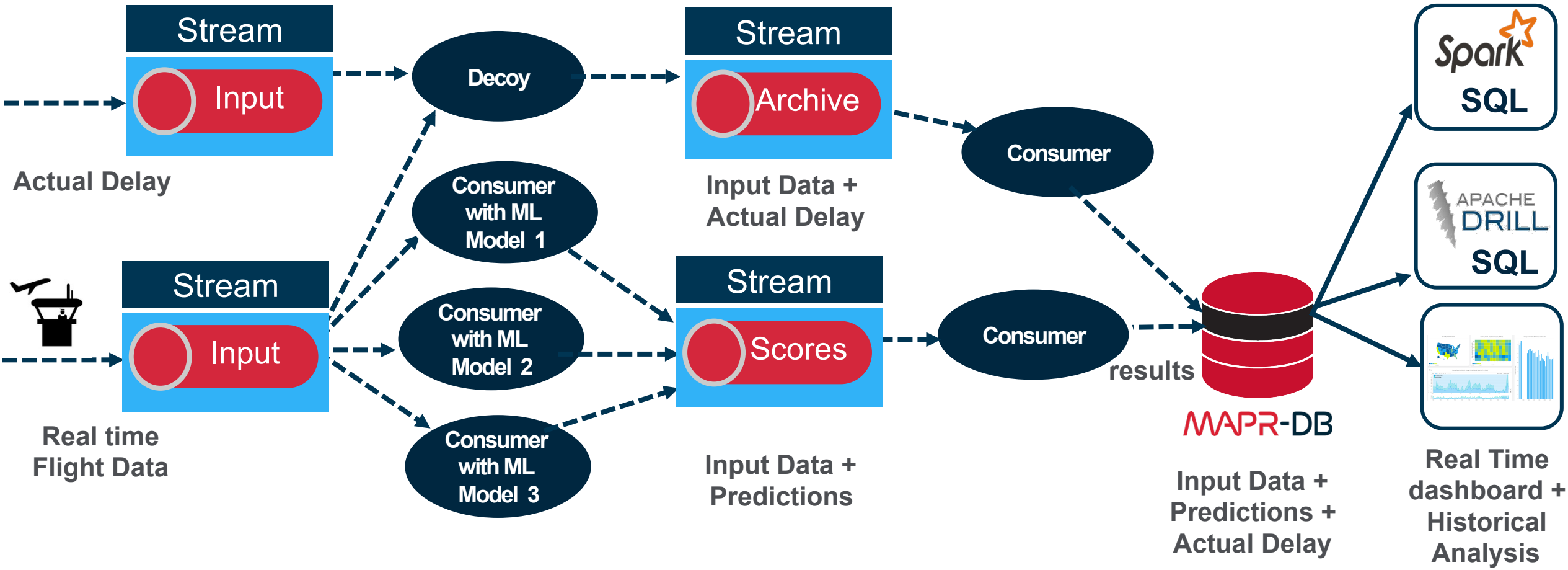
End to End Application Architecture



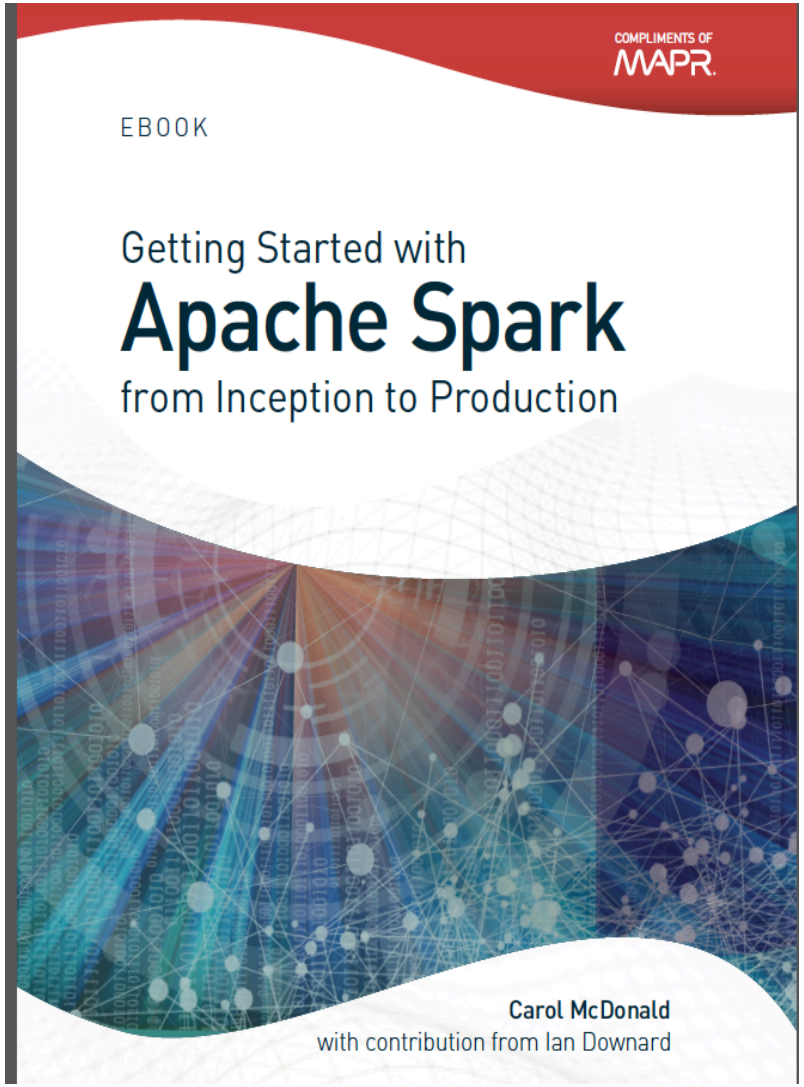
Simplified Pipeline: Real-Time Flight Delay Predictions



Machine Learning Pipeline for Flight Delays



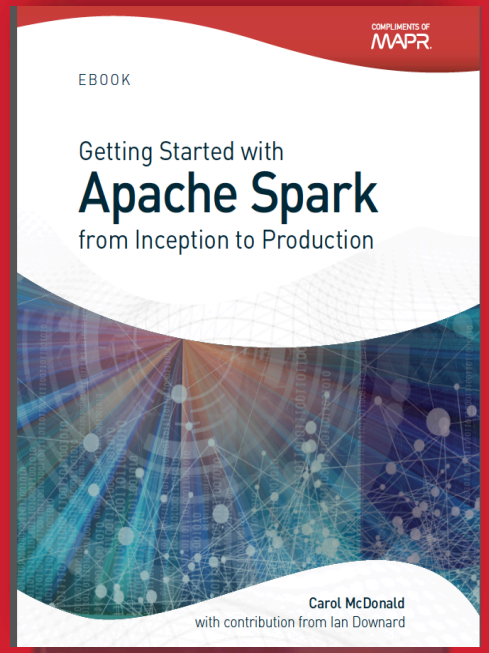
New Spark Ebook

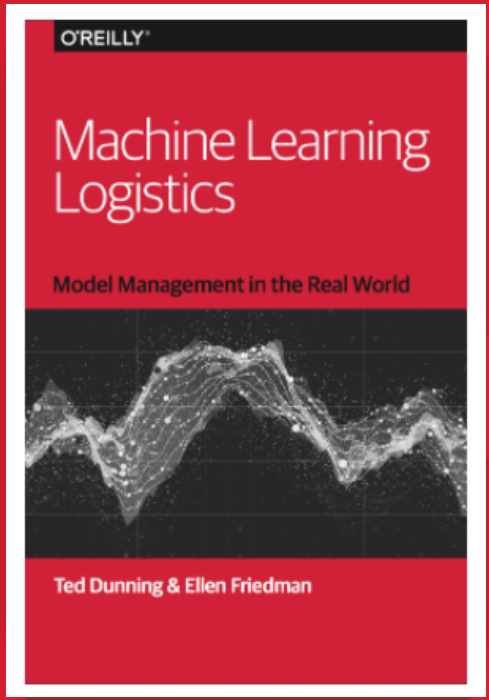


Link to Code for this webinar is in appendix of this Book.
https://mapr.com/ebooks/

MAPR

Complimentary books



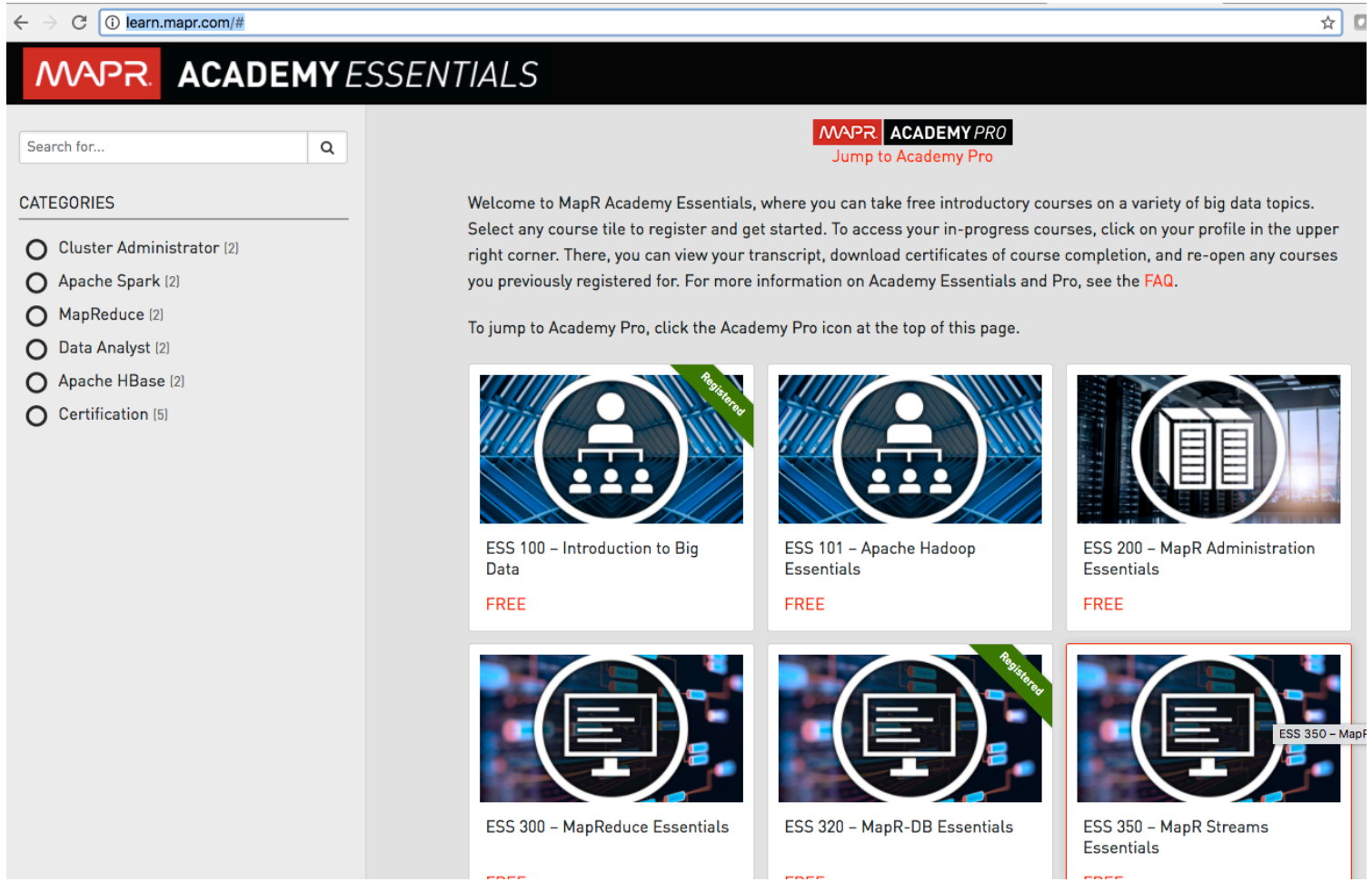


Get your copies

download at mapr.com/ebooks

To Learn More: New Spark 2.0 training

- **MapR Free ODT http://learn.mapr.com/**



MapR Blog

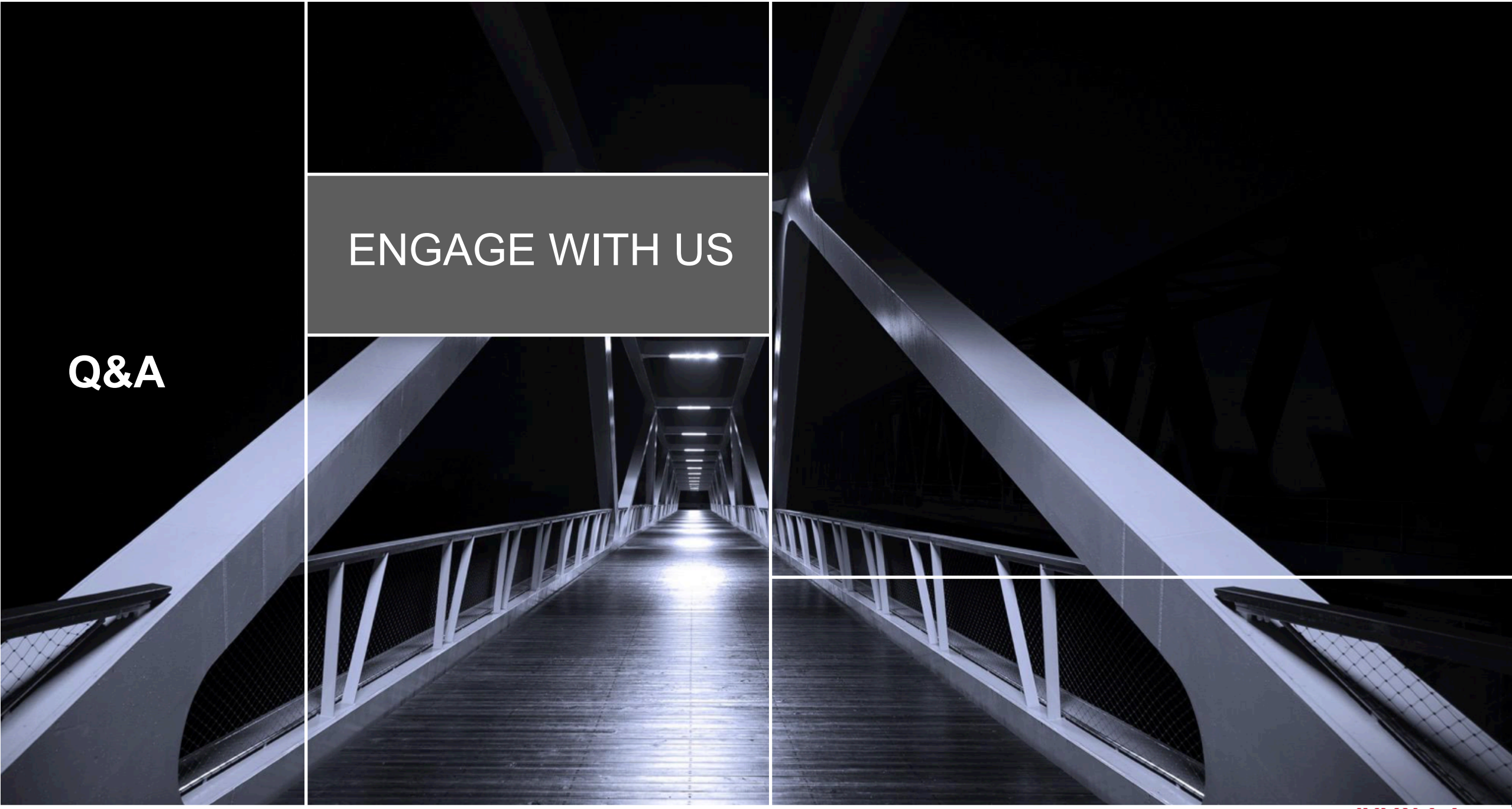
HTTPS://MAPR.COM/BLOG/

Featured Author

Carol McDonald

SOLUTIONS ARCHITECT, MAPR

Carol has extensive experience as a developer and architect building complex mission critical applications in the Banking, Health insurance and Telecom Industries. As a Java Technology Evangelist at Sun Microsystems, Carol traveled all over the world speaking at Sun Tech Days, JUGs, Companies, and Conferences. She is a recognized speaker in Java communities.



ENGAGE WITH US

Q&A