Decision Trees in Apache Spark

By

Max Karl

Hafiz Mujadid Khalid

Today's Agenda

- Intro to classification
- Decision Tree
- Intro to Apache Spark ML
- Flight delay prediction
  - Explain data
  - Explain code
  - Explain results

Supervised Learning

- Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.

$$Y = f(X)$$

- The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

Supervised Learning

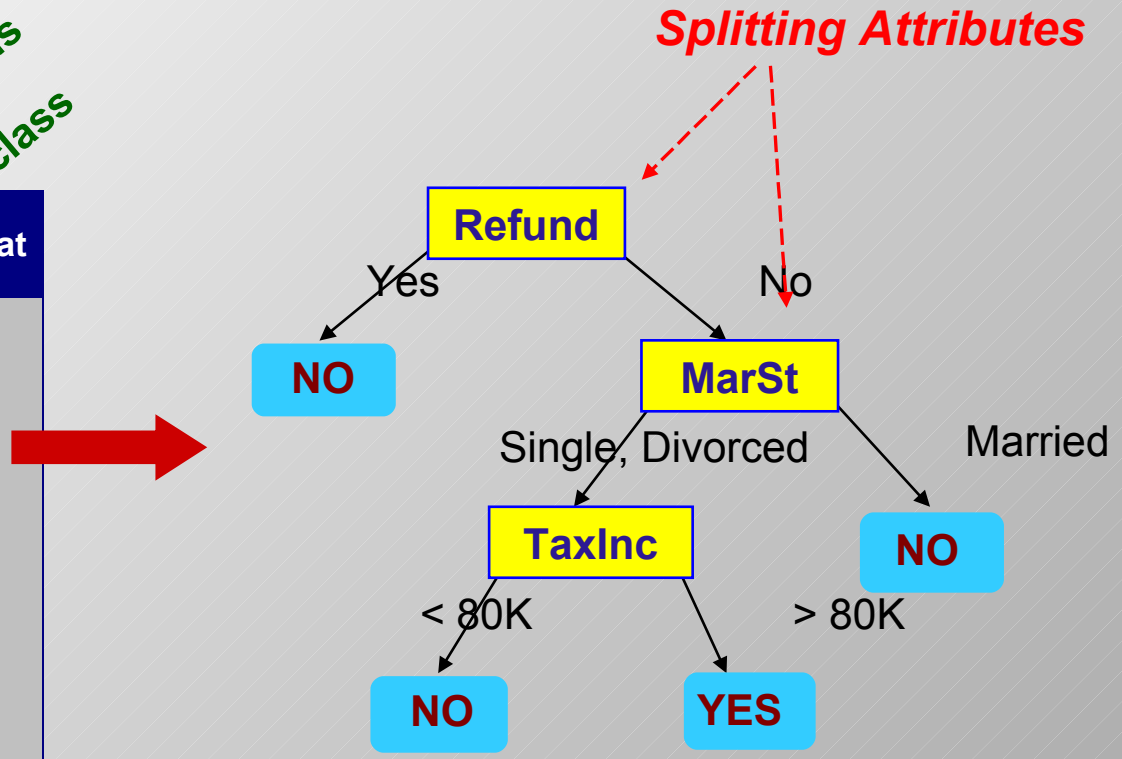
Supervised learning problems can be further grouped into regression and classification problems.

- **Classification**: Label denotes some categorical class like rainy, sunny, cloudy
- **Regression**: Label donates real value like income or house price

Decision Tree

Tid	Refund	Marital Status	Taxable Income	Cheat
		categorical	categorical	continuous	class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data



Model: Decision Tree

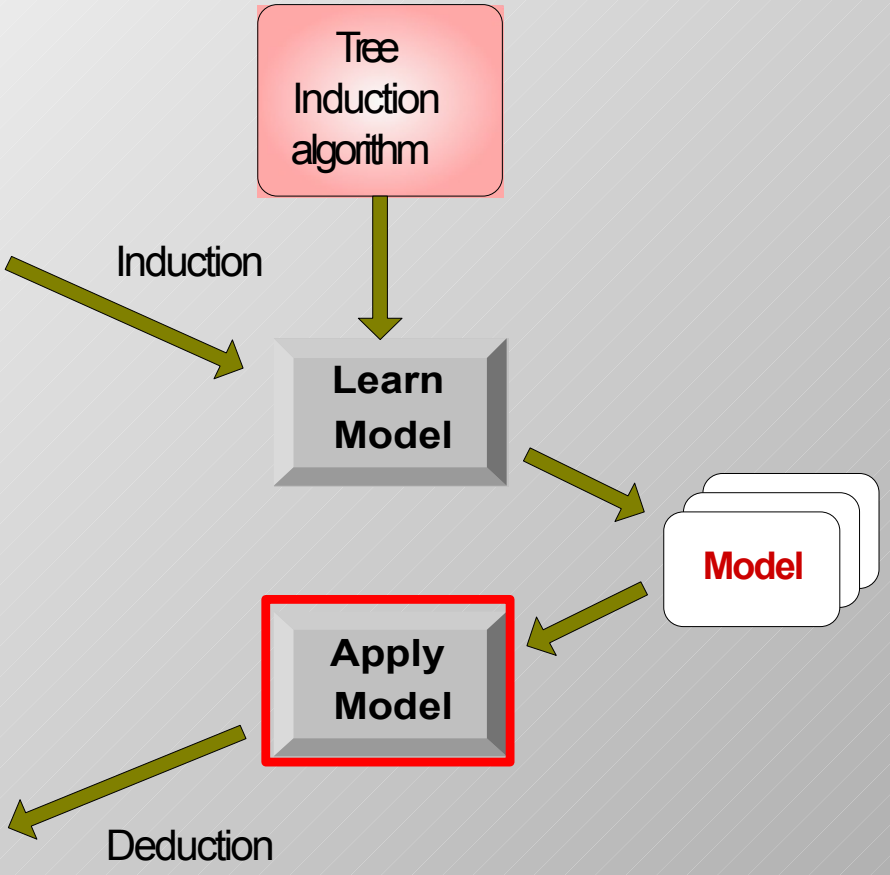
Decision Tree Classification

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

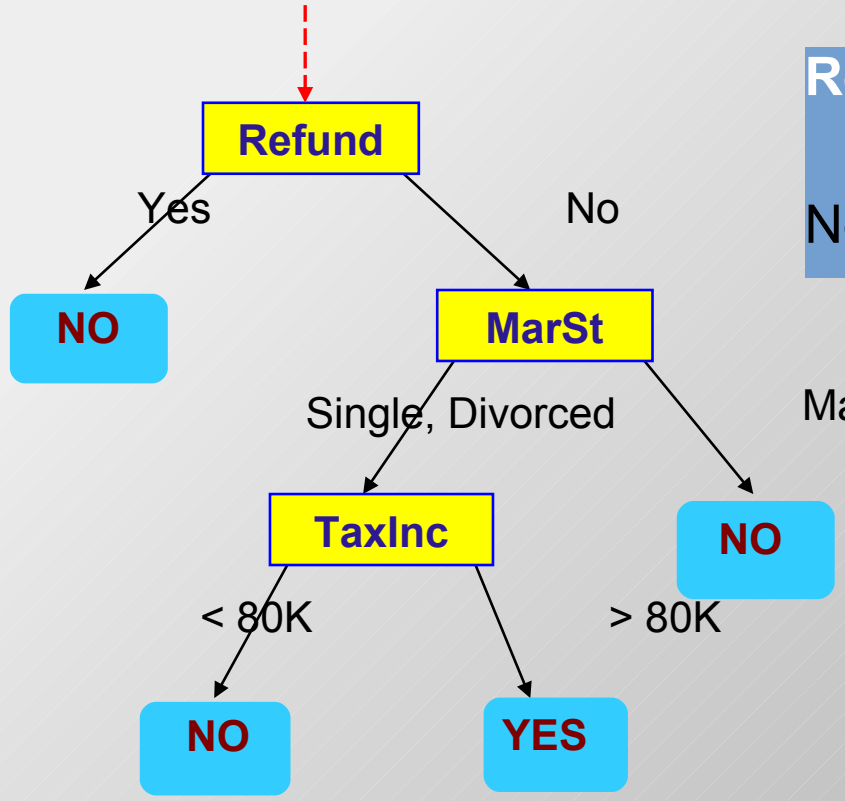
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Apply model on Test Data

Start from the root of tree.



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Decision Tree Induction

Hunt's Algorithm (one of the earliest)

Let D_t be the set of training records that reach a node t

General Procedure:

If D_t contains records that belong the same class y_t,

then t is a leaf node labeled as y_t

If D_t is an empty set,

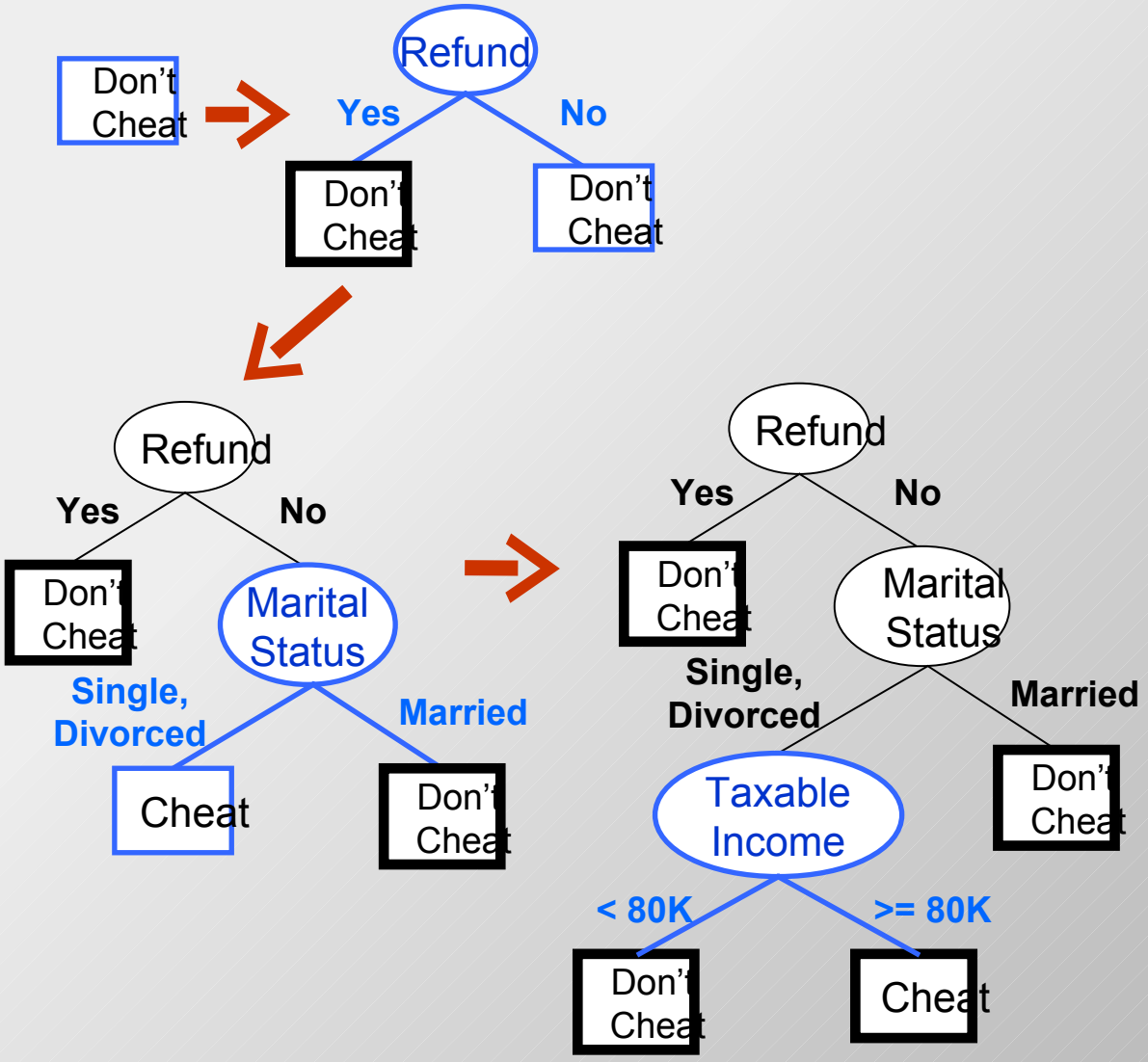
then t is a leaf node labeled by the default class, y_d

If D_t contains records that belong to more than one class,

use an attribute test to split the data into smaller subsets.

Recursively apply the procedure to each subset.

Hunt's Algorithm



Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

How to determine the Best Split

- Greedy approach
  - Nodes with homogeneous class distribution are preferred
- Need a measure of node impurity
  -

C0:5 C1:5	C0:9 C1:1
Non-homogeneous, **High degree of impurity**	**Homogeneous,** **Low degree of impurity**

Measures of Node Impurity

- Gini Index
- Entropy
- Misclassification error

Measure of Impurity: GINI

- Gini Index for a given node t :

$$GINI(t) = 1 - \sum_j [p(j|t)]^2$$

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

  - Maximum (1 - $1/n_c$) when records are equally distributed among all classes, implying least interesting information
  - Minimum (0.0) when all records belong to one class, implying most interesting information

C1	**0**
C2	**6**
Gini=0.000	

C1	**1**
C2	**5**
Gini=0.278	

C1	**2**
C2	**4**
Gini=0.444	

C1	**3**
C2	**3**
Gini=0.500	

Entropy

- Entropy at a given node t:

$$Entropy(t) = -\sum_j p(j|t) \log p(j|t)$$

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

  - Measures homogeneity of a node.
    - Maximum ($\log n_c$) when records are equally distributed among all classes implying least information
    - Minimum (0.0) when all records belong to one class, implying most information
  - Entropy based computations are similar to the GINI index computations

Examples for computing Entropy

$$Entropy(t) = -\sum_j p(j|t) \log_2 p(j|t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Classification Error

Classification error at a node t :

$$Error(t) = 1 - \max_i P(i|t)$$

Measures misclassification error made by a node.

- Maximum (1 - 1/n_c) when records are equally distributed among all classes, implying least interesting information
- Minimum (0.0) when all records belong to one class, implying most interesting information

Examples for Computing Error

$$Error(t) = 1 - \max_i P(i \mid t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Error = 1 – max (0, 1) = 1 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Advantages of Decision Trees?

- Easy to interpret
- Handle categorical variables
- (Multi-class) classification and regression
- No feature scaling
- Capture non-linearities and feature interactions
- Handle missing values

Disadvantages of Decision Trees?

- Time Complexity
- Over-fitting
- Instability - The decision tree changes by perturbing the dataset a bit. This is not desirable as we want our classification algorithm to be pretty robust to noise and be able to generalize well to future observed data.
- Decision Tree's do not work best if you have a lot of un-correlated variables

Spark's Machine Learning Library

- Consists of common learning algorithms and utilities, including:
  - Classification
  - Regression
  - Clustering
  - Collaborative filtering
  - Dimensionality reduction
- Two packages
  - spark.mllib
  - spark.ml

ML: Transformer

- A Transformer is a class which can transform one DataFrame into another DataFrame
- A Transformer implements transform()
  - Examples
  - HashingTF
  - Bucketizer
  - LogisticRegressionModel
  - PipelineModel
  - Binarizer

ML: Estimator

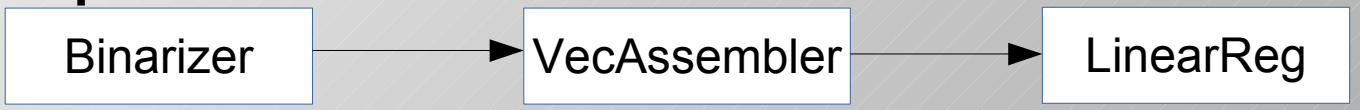
- An Estimator is a class which can take a DataFrame and produce a Transformer
- An Estimator implements fit()
- Examples
  - RandomForestClassifier
  - CrossValidator
  - IDF
  - Pipeline
  - StandardScaler

ML: Pipelines

A Pipeline is an estimator that contains stages representing a reusable work-flow Pipeline stages can be either estimators or transformers.

Pipeline

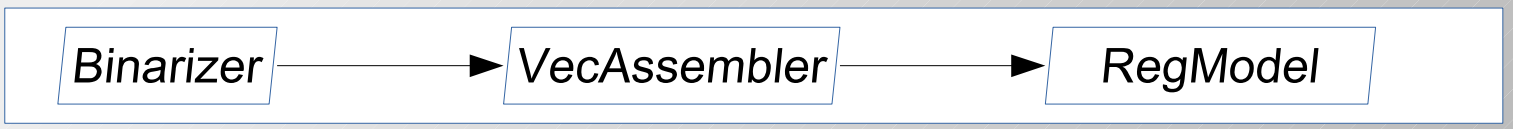
Pipeline



ML: PipelineModel

Train Model

PipelineModel



Classification with Spark ML

Datasource

- Flight data from Bureau of Transportation Statistics

Dataset

- All US flights from January to March 2017
- 1.35 Million flights
- 154063 flight with a delay of more than 40 minutes (11% of flights are delayed)

Classification with Spark ML

Data

DAY_OF_MONTH	DAY_OF_WEEK	CARRIER	TAIL_NUM	FL_NUM	ORIGIN	DEST	DEP_TIME	DEP_DELAY_	CRS_ARR_TIME	ARR_DELAY_NEW	DISTANCE
1	7	EV	N12563	4100	ORD	ABE	1855	0.00	2200	0.00	655.00
1	7	OO	N453SW	4515	DTW	ABE	1527	0.00	1658	0.00	425.00
1	7	EV	N153PQ	5333	ATL	ABE	2136	10.00	2326	17.00	692.00
2	1	OO	N439SW	4487	DTW	ABE	2057	50.00	2143	34.00	425.00
2	1	OO	N487CA	4493	DTW	ABE	1355	0.00	1526	2.00	425.00
2	1	EV	N200PQ	5267	ATL	ABE	1452	47.00	1604	32.00	692.00
2	1	EV	N228PQ	5333	ATL	ABE	2356	150.00	2328	126.00	692.00
2	1	EV	N12900	3981	ORD	ABE	1522	97.00	1640	97.00	655.00
3	2	EV	N27200	3981	ORD	ABE	1528	103.00	1640	86.00	655.00
3	2	EV	N16559	4100	ORD	ABE	1807	22.00	2040	21.00	655.00
3	2	EV	N607LR	5267	ATL	ABE	1435	30.00	1604	16.00	692.00
3	2	EV	N232PQ	5333	ATL	ABE	2121	0.00	2328	0.00	692.00
3	2	OO	N8828D	4487	DTW	ABE	2006	0.00	2143	0.00	425.00
3	2	OO	N440SW	4493	DTW	ABE	1353	0.00	1526	0.00	425.00
4	3	OO	N437SW	4487	DTW	ABE	2006	0.00	2143	18.00	425.00
4	3	OO	N426SW	4493	DTW	ABE	1351	0.00	1526	7.00	425.00
4	3	EV	N200PQ	5267	ATL	ABE	1400	0.00	1604	0.00	692.00
4	3	EV	N538CA	5333	ATL	ABE	2135	9.00	2328	5.00	692.00
4	3	EV	N14542	4100	ORD	ABE	1734	0.00	2040	0.00	655.00
5	4	EV	N14542	3981	ORD	ABE	1344	0.00	1642	0.00	655.00
5	4	EV	N21154	4100	ORD	ABE	2254	309.00	2040	301.00	655.00
5	4	EV	N232PQ	5267	ATL	ABE	1400	0.00	1604	0.00	692.00
5	4	EV	N132EV	5333	ATL	ABE	2120	0.00	2328	0.00	692.00
5	4	OO	N910EV	4487	DTW	ABE	2006	0.00	2143	0.00	425.00
5	4	OO	N453SW	4493	DTW	ABE	1404	8.00	1526	5.00	425.00
6	5	EV	N134EV	5267	ATL	ABE	1409	4.00	1604	0.00	692.00

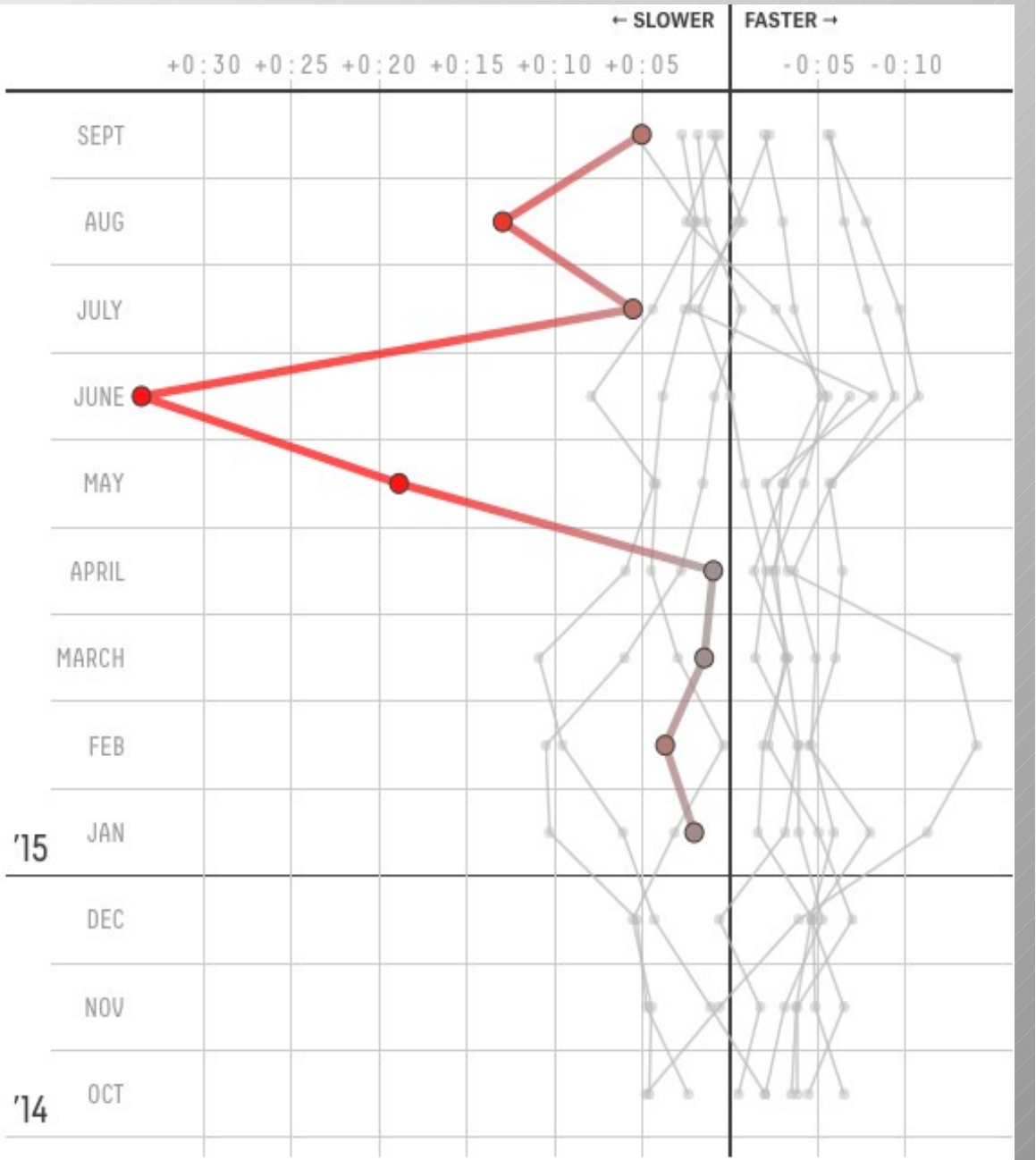
Classification with Spark ML

The Fastest Airlines

Each major airline ranked by the time it shaved off a typical flight relative to other airlines, October 2014 through September 2015

1	Virgin	-0:07
2	Alaska	-0:06
3	Delta	-0:04
4	Hawaiian	-0:03
5	US Airways*	-0:03
6	JetBlue	-0:03
7	Southwest	-0:01
8	American	+0:03
9	Frontier	+0:04
10	United	+0:04
11	**Spirit**	**+0:09**

*Because of its ongoing merger with American Airlines, US Airways' flights were classified as American Airlines flights starting in July 2015.



Classification with Spark ML

The problem

- Use given data
- Predict if flights will be delayed or not
- Measure the error rate of the prediction

→ use Databricks notebook to share program

Classification with Spark ML

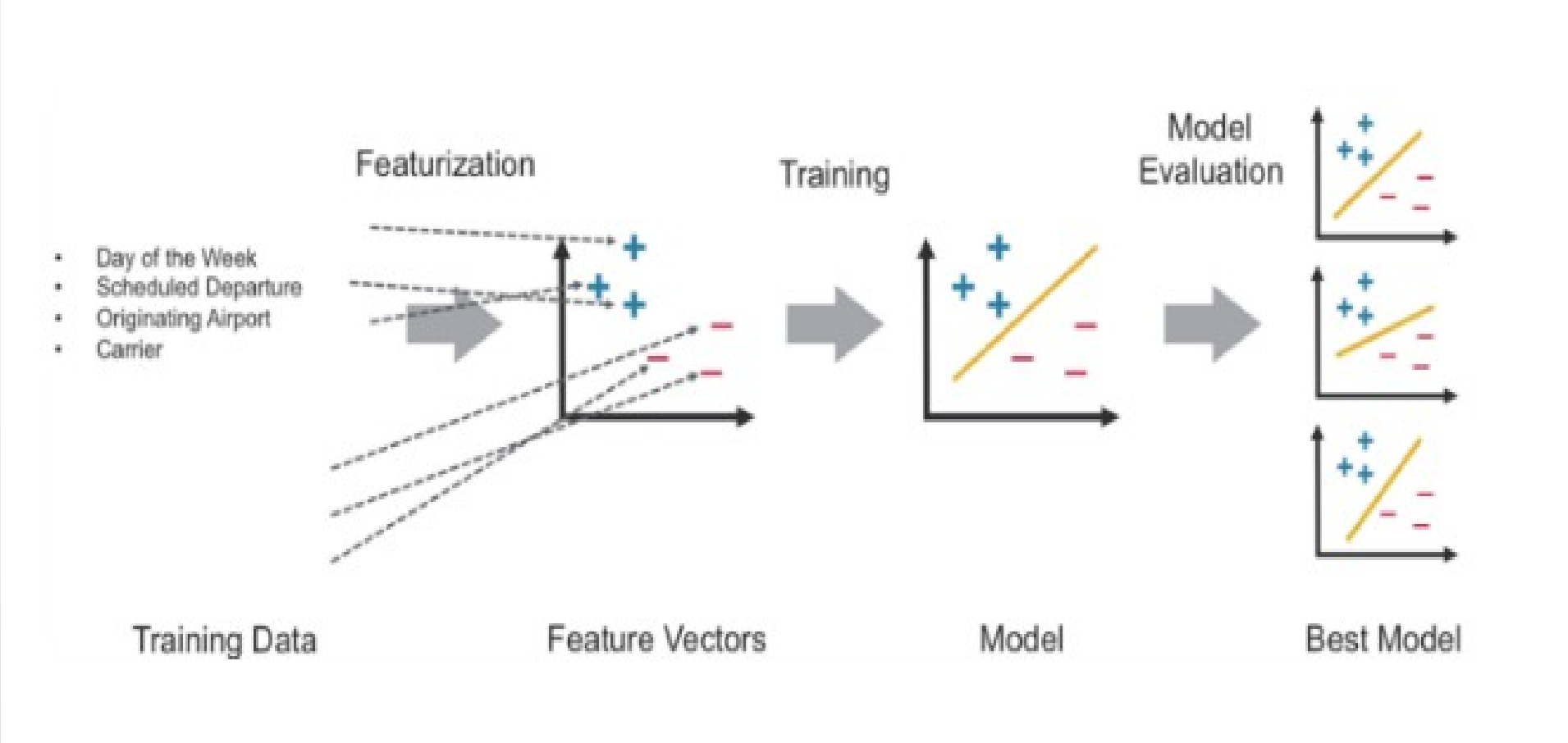
Code

Pre-processing

```
from pyspark.sql.functions import col  # for indicating a column using a string in the line below
from pyspark.sql.functions import udf
from pyspark.sql.types import *
def toID (value):
  return ''.join(str(ord(a)) for a in value)

udftoID = udf(toID, StringType())
df = df.withColumn("CARRIERID", udftoID("CARRIER"))

df = df.na.drop()

df = df.select([col(c).cast("double").alias(c) for c in
df.select("DAY_OF_MONTH","DAY_OF_WEEK","ORIGIN_AIRPORT_ID","DEST_AIRPORT_ID","CRS_DEP_TIME","DEP_DELAY_NEW","CRS_ARR_TIME","CRS_ELAPSED_TIME",
"CARRIERID").columns])
def delayed (value):
   if value > 40 :
      return 1.0
   return 0.0

udfdelayed = udf(delayed, DoubleType())
df = df.withColumn("DELAYED", udfdelayed("DEP_DELAY_NEW"))
```

Classification with Spark ML

Define features array



Classification with Spark ML

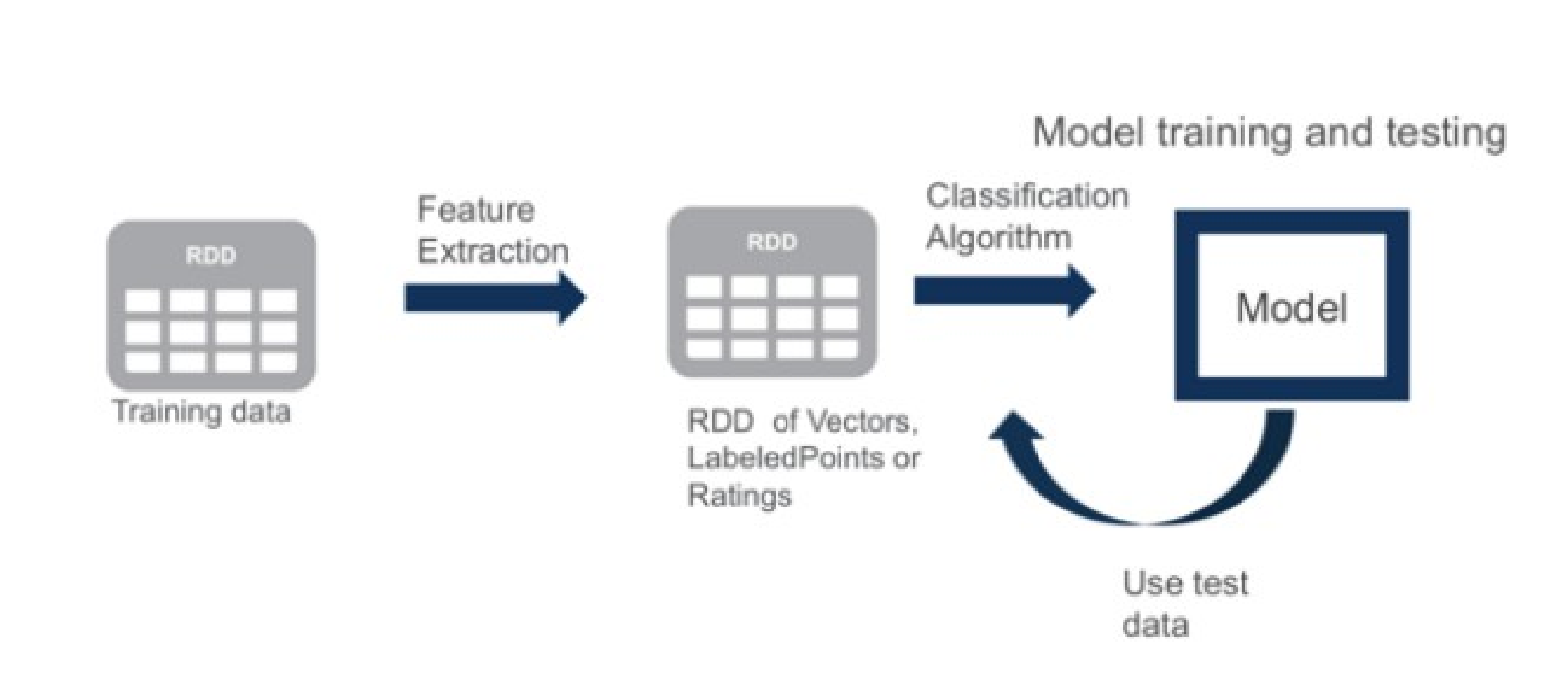
Code

Split data into traning and testing

```
# Split the dataset randomly into 75% for training and 25% for testing.
train, test = df.randomSplit([0.75, 0.25])
# print "We have %d training examples and %d test examples." % (train.count(), test.count())

from pyspark.ml.feature import VectorAssembler, VectorIndexer
featuresCols = df.columns
# remove labeled data from the features
featuresCols.remove('DELAYED')
featuresCols.remove('DEP_DELAY_NEW')

# This concatenates all feature columns into a single feature vector in a new column "rawFeatures".
vectorAssembler = VectorAssembler(inputCols=featuresCols, outputCol="rawFeatures")

# This identifies categorical features and indexes them.
vectorIndexer = VectorIndexer(inputCol="rawFeatures", outputCol="features", maxCategories=4)

```

Classification with Spark ML

Model layout



Classification with Spark ML

Build pipeline and train & evaluate model

```
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml import Pipeline
dt = DecisionTreeClassifier(labelCol="DELAYED", featuresCol="features",impurity='entropy')
pipeline = Pipeline(stages=[vectorAssembler, vectorIndexer, dt])
pipelineModel = pipeline.fit(train)
predictions = pipelineModel.transform(test)


from pyspark.ml.evaluation import BinaryClassificationEvaluator

# Evaluate model
evaluator = BinaryClassificationEvaluator(labelCol="DELAYED" , rawPredictionCol="rawPrediction")
evaluator.evaluate(predictions)
```

Classification with Spark ML

Results

DELAYED	prediction	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN_AIRPORT_ID	DEST_AIRPORT_ID	CRS_DEP_TIME	CRS_ARR_TIME	CRS_ELAPSED_TIME	CARRIERID
0.0	0.0	1.0	7.0	10136.0	11298.0	1248.0	1345.0	57.0	6986.0
0.0	0.0	1.0	7.0	10140.0	11259.0	635.0	915.0	100.0	8778.0
0.0	0.0	1.0	7.0	10140.0	11259.0	1045.0	1325.0	100.0	8778.0
1.0	0.0	1.0	7.0	10140.0	11259.0	2025.0	2255.0	90.0	8778.0
0.0	0.0	1.0	7.0	10140.0	11292.0	545.0	710.0	85.0	8565.0
0.0	0.0	1.0	7.0	10140.0	11298.0	500.0	749.0	109.0	6565.0
0.0	0.0	1.0	7.0	10140.0	11298.0	800.0	1047.0	107.0	6565.0
0.0	0.0	1.0	7.0	10140.0	11298.0	1637.0	1922.0	105.0	6565.0
0.0	0.0	1.0	7.0	10140.0	12191.0	630.0	930.0	120.0	8778.0
0.0	0.0	1.0	7.0	10140.0	12191.0	1605.0	1905.0	120.0	8778.0
0.0	0.0	1.0	7.0	10140.0	12266.0	1110.0	1415.0	125.0	6986.0
0.0	0.0	1.0	7.0	10140.0	12266.0	1705.0	2010.0	125.0	6986.0
0.0	0.0	1.0	7.0	10140.0	12478.0	2359.0	551.0	232.0	6654.0
0.0	0.0	1.0	7.0	10140.0	12889.0	1455.0	1535.0	100.0	8778.0
0.0	0.0	1.0	7.0	10140.0	12889.0	1630.0	1705.0	95.0	8778.0
0.0	0.0	1.0	7.0	10140.0	12892.0	731.0	842.0	131.0	7979.0
0.0	0.0	1.0	7.0	10140.0	13796.0	1340.0	1510.0	150.0	8778.0
0.0	0.0	1.0	7.0	10140.0	14057.0	1205.0	1410.0	185.0	8778.0
0.0	0.0	1.0	7.0	10140.0	14107.0	625.0	753.0	88.0	7979.0
0.0	0.0	1.0	7.0	10140.0	14107.0	1030.0	1150.0	80.0	8778.0
0.0	0.0	1.0	7.0	10140.0	14107.0	1520.0	1646.0	86.0	7979.0
1.0	1.0	1.0	7.0	10140.0	14747.0	1635.0	1845.0	190.0	6583.0
0.0	0.0	1.0	7.0	10158.0	11697.0	1955.0	2239.0	164.0	7875.0
0.0	0.0	1.0	7.0	10158.0	13204.0	600.0	836.0	156.0	7875.0

Classification with Spark ML

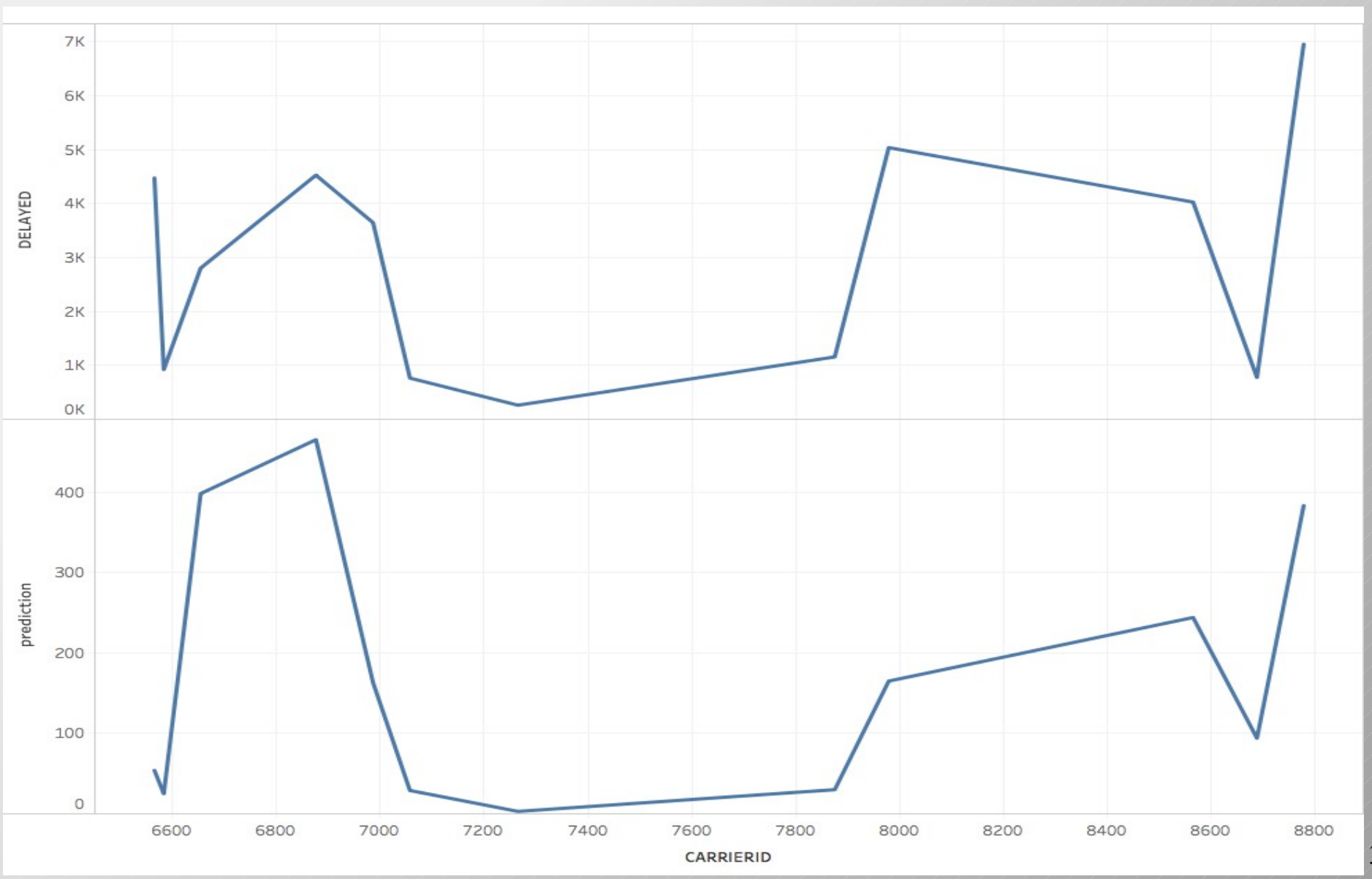
Results

- Small dataset (just January) 37% error rate
- Big dataset (January to March) 42% error rate

We got better results with GBT

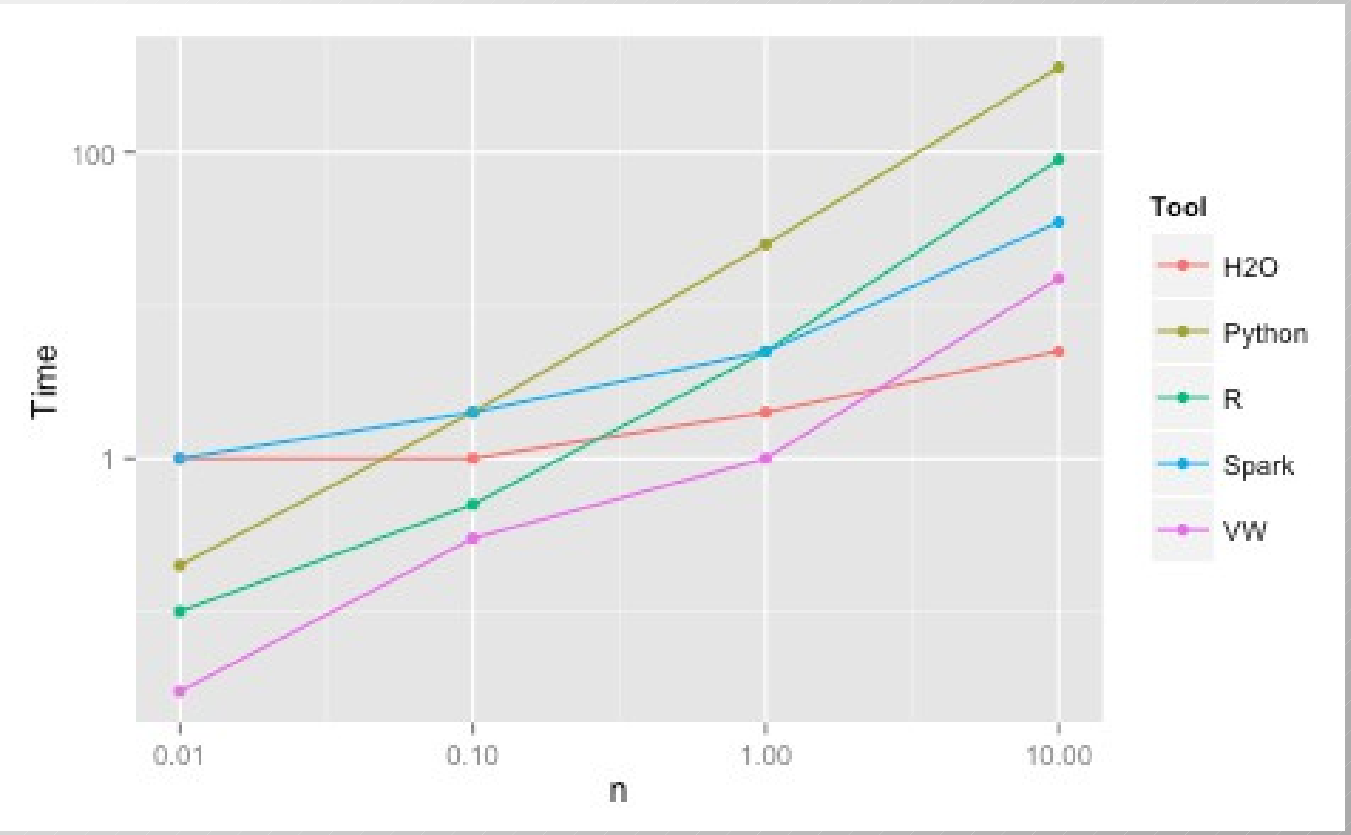
- Small dataset 10% error rate
- Big dataset 8,7% error rate
- Compared to 11% error rate if we just set all values to not delayed

Predicted and real values



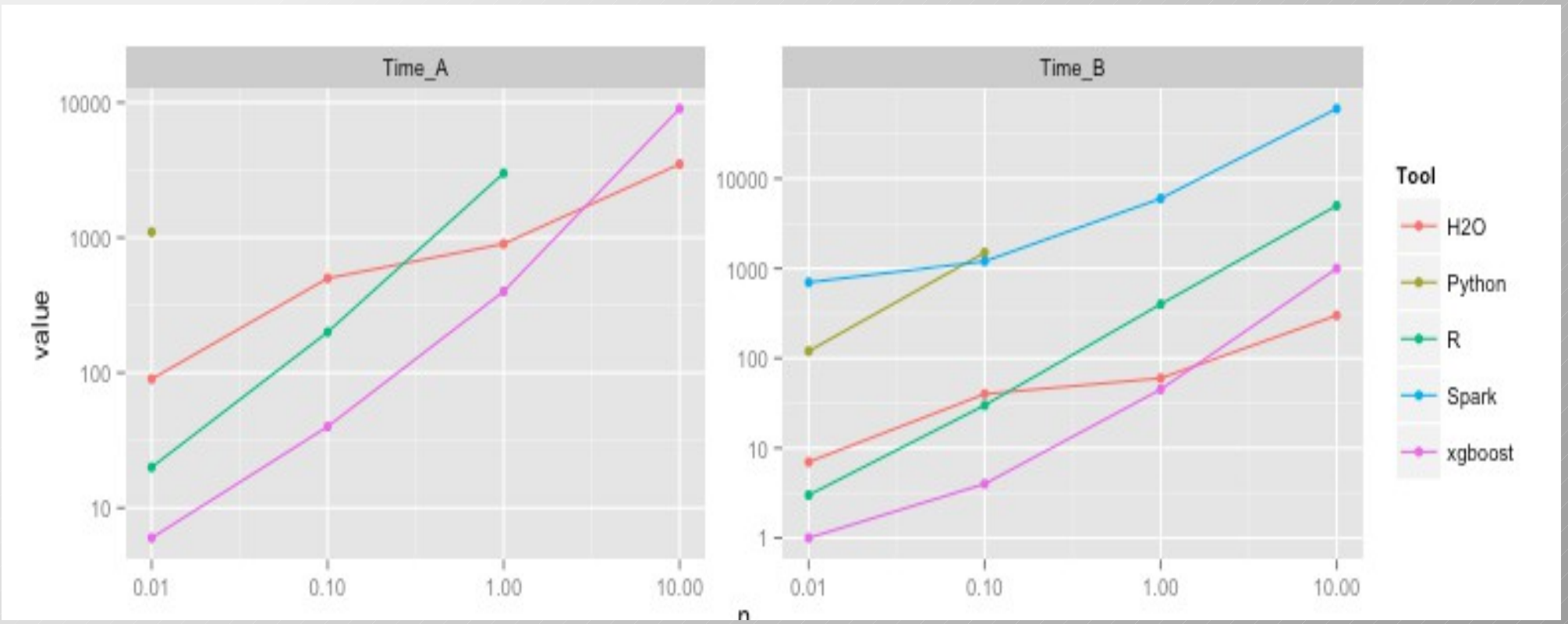
Scalability of Spark

n	Time (sec)	RAM (GB)	AUC
10K	1	1	66.6
100K	2	1	70.2
1M	5	2	70.9
10M	35	10	70.9



Scalability of Spark

n	Time (s) A	AUC A	RAM (GB) A	Time (s) B	AUC B	RAM (GB) B
10K	180000	66.4	30	700	67.8	10
100K	-	-	-	1200	72.3	30
1M	-	-	-	6000	73.8	30
10M	-	-	-	(60000)	(74.1)	crash



Conclusion

- Accuracy depends on classification algorithm
  - Affects the runtime
- Features influence the accuracy

References

- http://www-users.cs.umn.edu/~kumar/dmbook/index.php
- https://spark-summit.org/2014/scalable-distributed-decision-trees-in-spark-mllib/
- https://spark.apache.org/docs/2.0.2/ml-classification-regression.html#classification
- https://docs.databricks.com/spark/latest/mllib/decision-trees.html#decision-trees
- https://mapr.com/blog/apache-spark-machine-learning-tutorial/
- https://projects.fivethirtyeight.com/flights/
- https://github.com/szilard/benchm-ml

Thank you for your attention
Questions?