Machine Learning at Scale:

A Customer Use Case: Apache Spark to the Rescue

SCV Data Science and Machine Learning Meetup • 10.16.2018

The Environment

Use Case: Churn Prevention

What is *customer churn?*

Customer churn is when an existing customer, user, player, subscriber or any kind of return client stops doing business or ends the relationship with a company.

Overview

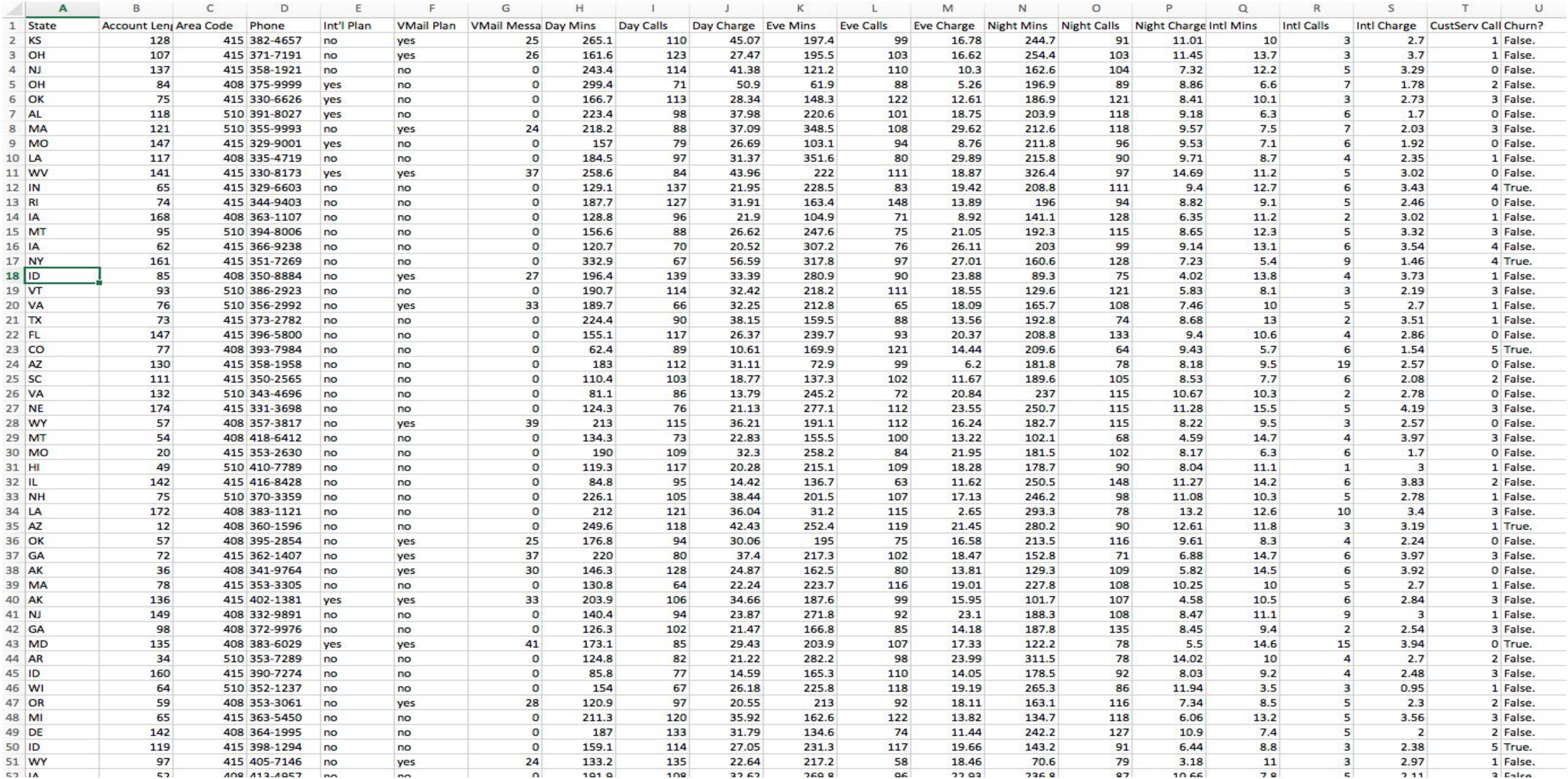
- Step 1: Interview Stakeholders to determine feature set.
- Step 2: Examine Dataset.
- Step 3: Ingest Churn Data to Spark Notebook.
- Step 4: Enrich the data to Get Additional Insights.
- Step 5: Exploratory Data Analysis.
- Step 6: Visualization.
- Step 7: Model Creation
- Step 8: Results Interpretation

Step 1 - Interview Stakeholders to design feature set

Example: Verizon Digital Media Services POC ML Retention

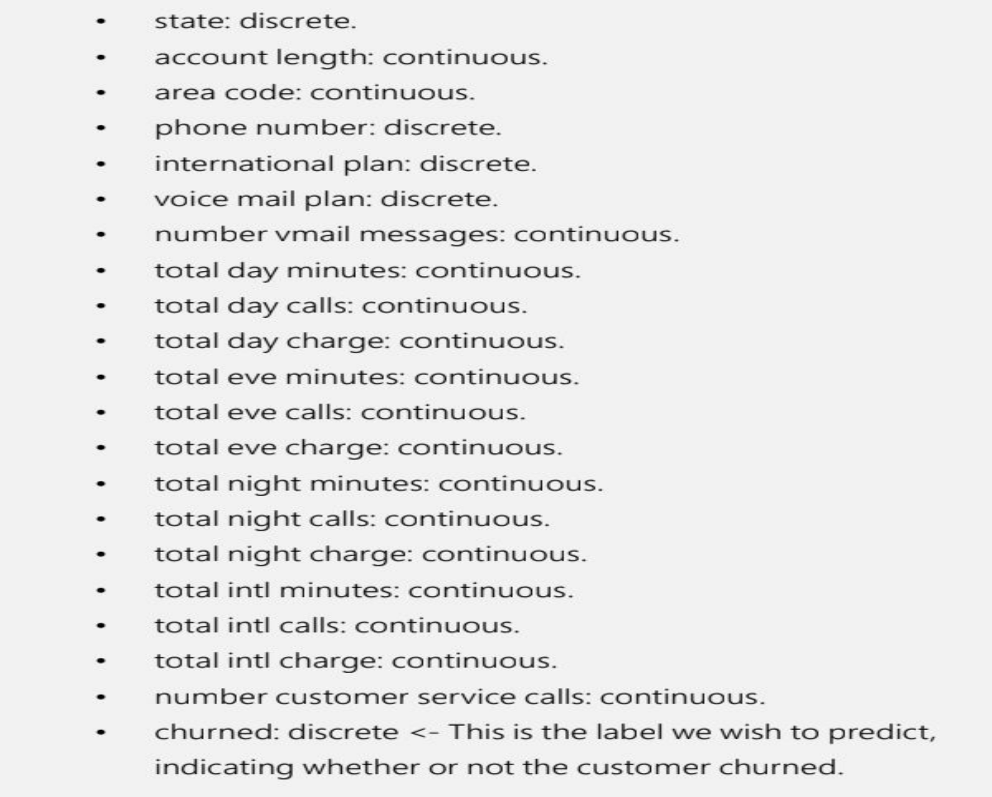
- Disparate Data Sources
- Define “churned” Customer, Timeframe, Active vs InActive, Create Boolean Field
- Discuss Possible Feature to include in training dataset: Number of active months, Monthly Usage, Products, Contract Type, ServiceNow Priority Tickets, Avg Time to Resolution, Revenue, Customer Recon (Salesforce and Google: Size, Industry)
- Survival Analysis is another method of predicting churn
- For this demo, we will use a publicly available dataset from the uci repository: churn.csv.

Step 2 - Examine Dataset



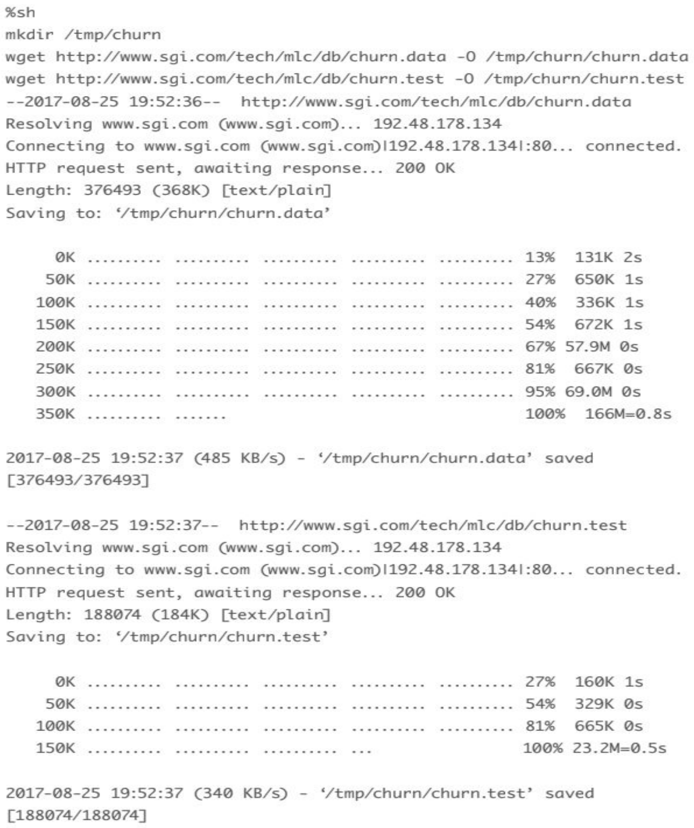
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	State	Account Len	Area Code	Phone	Int'l Plan	VMail Plan	VMail Messa	Day Mins	Day Calls	Day Charge	Eve Mins	Eve Calls	Eve Charge	Night Mins	Night Calls	Night Charge	Intl Mins	Intl Calls	Intl Charge	CustServ Call	Churn?
2	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	197.4	99	16.78	244.7	91	11.01	10	3	2.7	1	False.
3	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	195.5	103	16.62	254.4	103	11.45	13.7	3	3.7	1	False.
4	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	121.2	110	10.3	162.6	104	7.32	12.2	5	3.29	0	False.
5	OH	84	408	375-9999	yes	no	0	299.4	71	50.9	61.9	88	5.26	196.9	89	8.86	6.6	7	1.78	2	False.
6	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	148.3	122	12.61	186.9	121	8.41	10.1	3	2.73	3	False.
7	AL	118	510	391-8027	yes	no	0	223.4	98	37.98	220.6	101	18.75	203.9	118	9.18	6.3	6	1.7	0	False.
8	MA	121	510	355-9993	no	yes	24	218.2	88	37.09	348.5	108	29.62	212.6	118	9.57	7.5	7	2.03	3	False.
9	MO	147	415	329-9001	yes	no	0	157	79	26.69	103.1	94	8.76	211.8	96	9.53	7.1	6	1.92	0	False.
10	LA	117	408	335-4719	no	no	0	184.5	97	31.37	351.6	80	29.89	215.8	90	9.71	8.7	4	2.35	1	False.
11	WV	141	415	330-8173	yes	yes	37	258.6	84	43.96	222	111	18.87	326.4	97	14.69	11.2	5	3.02	0	False.
12	IN	65	415	329-6603	no	no	0	129.1	137	21.95	228.5	83	19.42	208.8	111	9.4	12.7	6	3.43	4	True.
13	RI	74	415	344-9403	no	no	0	187.7	127	31.91	163.4	148	13.89	196	94	8.82	9.1	5	2.46	0	False.
14	IA	168	408	363-1107	no	no	0	128.8	96	21.9	104.9	71	8.92	141.1	128	6.35	11.2	2	3.02	1	False.
15	MT	95	510	394-8006	no	no	0	156.6	88	26.62	247.6	75	21.05	192.3	115	8.65	12.3	5	3.32	3	False.
16	IA	62	415	366-9238	no	no	0	120.7	70	20.52	307.2	76	26.11	203	99	9.14	13.1	6	3.54	4	False.
17	NY	161	415	351-7269	no	no	0	332.9	67	56.59	317.8	97	27.01	160.6	128	7.23	5.4	9	1.46	4	True.
18	ID	85	408	350-8884	no	yes	27	196.4	139	33.39	280.9	90	23.88	89.3	75	4.02	13.8	4	3.73	1	False.
19	VT	93	510	386-2923	no	no	0	190.7	114	32.42	218.2	111	18.55	129.6	121	5.83	8.1	3	2.19	3	False.
20	VA	76	510	356-2992	no	yes	33	189.7	66	32.25	212.8	65	18.09	165.7	108	7.46	10	5	2.7	1	False.
21	TX	73	415	373-2782	no	no	0	224.4	90	38.15	159.5	88	13.56	192.8	74	8.68	13	2	3.51	1	False.
22	FL	147	415	396-5800	no	no	0	155.1	117	26.37	239.7	93	20.37	208.8	133	9.4	10.6	4	2.86	0	False.
23	CO	77	408	393-7984	no	no	0	62.4	89	10.61	169.9	121	14.44	209.6	64	9.43	5.7	6	1.54	5	True.
24	AZ	130	415	358-1958	no	no	0	183	112	31.11	72.9	99	6.2	181.8	78	8.18	9.5	19	2.57	0	False.
25	SC	111	415	350-2565	no	no	0	110.4	103	18.77	137.3	102	11.67	189.6	105	8.53	7.7	6	2.08	2	False.
26	VA	132	510	343-4696	no	no	0	81.1	86	13.79	245.2	72	20.84	237	115	10.67	10.3	2	2.78	0	False.
27	NE	174	415	331-3698	no	no	0	124.3	76	21.13	277.1	112	23.55	250.7	115	11.28	15.5	5	4.19	3	False.
28	WY	57	408	357-3817	no	yes	39	213	115	36.21	191.1	112	16.24	182.7	115	8.22	9.5	3	2.57	0	False.
29	MT	54	408	418-6412	no	no	0	134.3	73	22.83	155.5	100	13.22	102.1	68	4.59	14.7	4	3.97	3	False.
30	MO	20	415	353-2630	no	no	0	190	109	32.3	258.2	84	21.95	181.5	102	8.17	6.3	6	1.7	0	False.
31	HI	49	510	410-7789	no	no	0	119.3	117	20.28	215.1	109	18.28	178.7	90	8.04	11.1	1	3	1	False.
32	IL	142	415	416-8428	no	no	0	84.8	95	14.42	136.7	63	11.62	250.5	148	11.27	14.2	6	3.83	2	False.
33	NH	75	510	370-3359	no	no	0	226.1	105	38.44	201.5	107	17.13	246.2	98	11.08	10.3	5	2.78	1	False.
34	LA	172	408	383-1121	no	no	0	212	121	36.04	31.2	115	2.65	293.3	78	13.2	12.6	10	3.4	3	False.
35	AZ	12	408	360-1596	no	no	0	249.6	118	42.43	252.4	119	21.45	280.2	90	12.61	11.8	3	3.19	1	True.
36	OK	57	408	395-2854	no	yes	25	176.8	94	30.06	195	75	16.58	213.5	116	9.61	8.3	4	2.24	0	False.
37	GA	72	415	362-1407	no	yes	37	220	80	37.4	217.3	102	18.47	152.8	71	6.88	14.7	6	3.97	3	False.
38	AK	36	408	341-9764	no	yes	30	146.3	128	24.87	162.5	80	13.81	129.3	109	5.82	14.5	6	3.92	0	False.
39	MA	78	415	353-3305	no	no	0	130.8	64	22.24	223.7	116	19.01	227.8	108	10.25	10	5	2.7	1	False.
40	AK	136	415	402-1381	yes	yes	33	203.9	106	34.66	187.6	99	15.95	101.7	107	4.58	10.5	6	2.84	3	False.
41	NJ	149	408	332-9891	no	no	0	140.4	94	23.87	271.8	92	23.1	188.3	108	8.47	11.1	9	3	1	False.
42	GA	98	408	372-9976	no	no	0	126.3	102	21.47	166.8	85	14.18	187.8	135	8.45	9.4	2	2.54	3	False.
43	MD	135	408	383-6029	yes	yes	41	173.1	85	29.43	203.9	107	17.33	122.2	78	5.5	14.6	15	3.94	0	True.
44	AR	34	510	353-7289	no	no	0	124.8	82	21.22	282.2	98	23.99	311.5	78	14.02	10	4	2.7	2	False.
45	ID	160	415	390-7274	no	no	0	85.8	77	14.59	165.3	110	14.05	178.5	92	8.03	9.2	4	2.48	3	False.
46	WI	64	510	352-1237	no	no	0	154	67	26.18	225.8	118	19.19	265.3	86	11.94	3.5	3	0.95	1	False.
47	OR	59	408	353-3061	no	yes	28	120.9	97	20.55	213	92	18.11	163.1	116	7.34	8.5	5	2.3	2	False.
48	MI	65	415	363-5450	no	no	0	211.3	120	35.92	162.6	122	13.82	134.7	118	6.06	13.2	5	3.56	3	False.
49	DE	142	408	364-1995	no	no	0	187	133	31.79	134.6	74	11.44	242.2	127	10.9	7.4	5	2	2	False.
50	ID	119	415	398-1294	no	no	0	159.1	114	27.05	231.3	117	19.66	143.2	91	6.44	8.8	3	2.38	5	True.
51	WY	97	415	405-7146	no	yes	24	133.2	135	22.64	217.2	58	18.46	70.6	79	3.18	11	3	2.97	1	False.
52	IA	52	408	413-4957	no	no	0	191.9	108	32.62	269.8	96	22.93	236.8	87	10.66	7.8	5	2.11	3	False.

Step 2 (cont.) - Examine Dataset

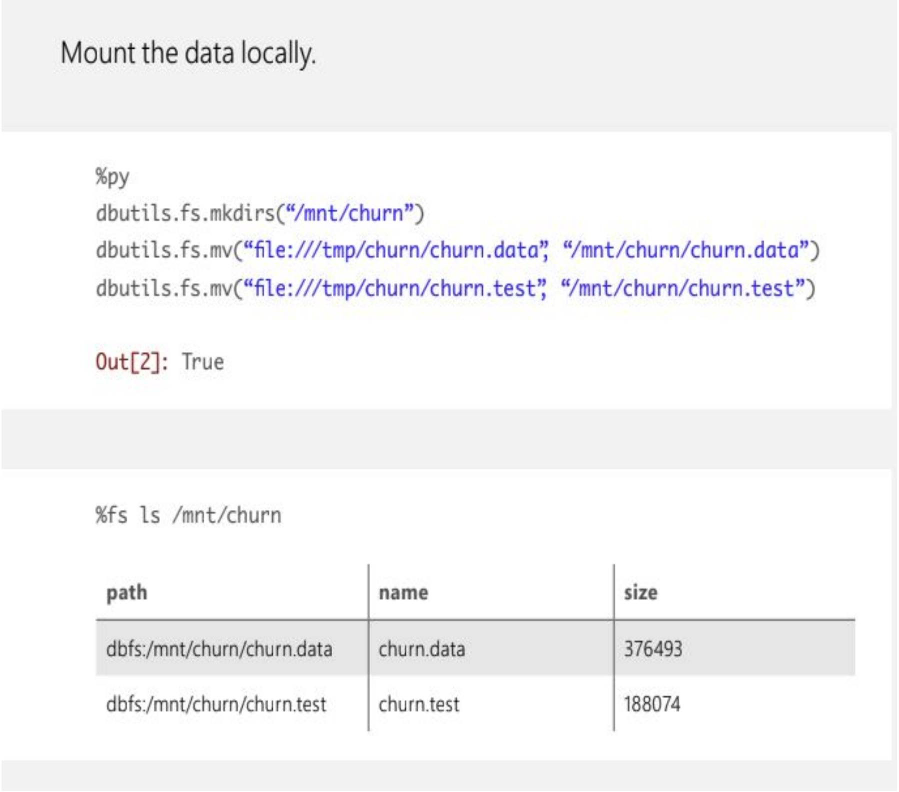


- state: discrete.
- account length: continuous.
- area code: continuous.
- phone number: discrete.
- international plan: discrete.
- voice mail plan: discrete.
- number vmail messages: continuous.
- total day minutes: continuous.
- total day calls: continuous.
- total day charge: continuous.
- total eve minutes: continuous.
- total eve calls: continuous.
- total eve charge: continuous.
- total night minutes: continuous.
- total night calls: continuous.
- total night charge: continuous.
- total intl minutes: continuous.
- total intl calls: continuous.
- total intl charge: continuous.
- number customer service calls: continuous.
- churned: discrete <- This is the label we wish to predict, indicating whether or not the customer churned.

Step 3: Ingest Churn Data to Spark Notebook



```
%sh
mkdir /tmp/churn
wget http://www.sgi.com/tech/mlc/db/churn.data -O /tmp/churn/churn.data
wget http://www.sgi.com/tech/mlc/db/churn.test -O /tmp/churn/churn.test
--2017-08-25 19:52:36--  http://www.sgi.com/tech/mlc/db/churn.data
Resolving www.sgi.com (www.sgi.com)... 192.48.178.134
Connecting to www.sgi.com (www.sgi.com)|192.48.178.134|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 376493 (368K) [text/plain]
Saving to: '/tmp/churn/churn.data'

     0K .......... .......... .......... .......... .......... 13%  131K 2s
    50K .......... .......... .......... .......... .......... 27%  650K 1s
   100K .......... .......... .......... .......... .......... 40%  336K 1s
   150K .......... .......... .......... .......... .......... 54%  672K 1s
   200K .......... .......... .......... .......... .......... 67% 57.9M 0s
   250K .......... .......... .......... .......... .......... 81%  667K 0s
   300K .......... .......... .......... .......... .......... 95% 69.0M 0s
   350K .......... .......                                    100%  166M=0.8s

2017-08-25 19:52:37 (485 KB/s) - '/tmp/churn/churn.data' saved
[376493/376493]

--2017-08-25 19:52:37--  http://www.sgi.com/tech/mlc/db/churn.test
Resolving www.sgi.com (www.sgi.com)... 192.48.178.134
Connecting to www.sgi.com (www.sgi.com)|192.48.178.134|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 188074 (184K) [text/plain]
Saving to: '/tmp/churn/churn.test'

     0K .......... .......... .......... .......... .......... 27%  160K 1s
    50K .......... .......... .......... .......... .......... 54%  329K 0s
   100K .......... .......... .......... .......... .......... 81%  665K 0s
   150K .......... .......... .......... ...                  100% 23.2M=0.5s

2017-08-25 19:52:37 (340 KB/s) - '/tmp/churn/churn.test' saved
[188074/188074]
```



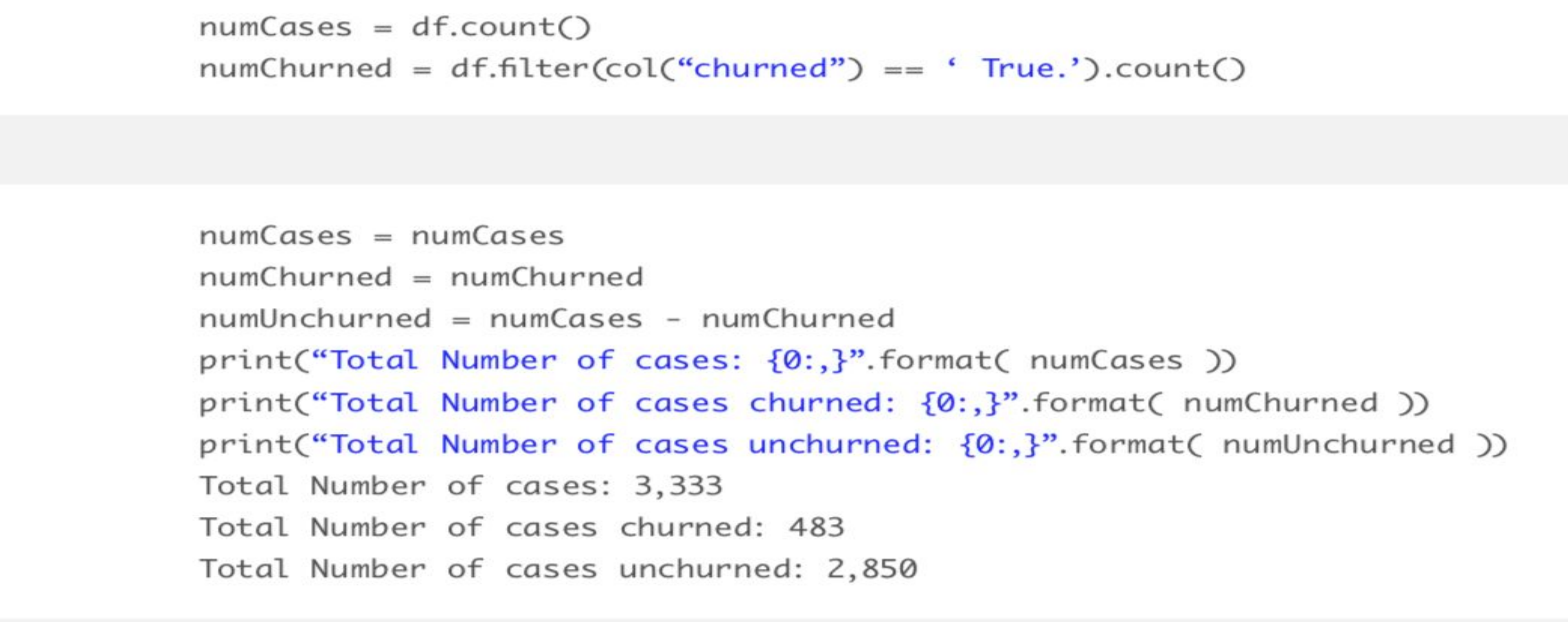
Mount the data locally.

```
%py
dbutils.fs.mkdirs("/mnt/churn")
dbutils.fs.mv("file:///tmp/churn/churn.data", "/mnt/churn/churn.data")
dbutils.fs.mv("file:///tmp/churn/churn.test", "/mnt/churn/churn.test")

Out[2]: True
```

```
%fs ls /mnt/churn
```

path	name	size
dbfs:/mnt/churn/churn.data	churn.data	376493
dbfs:/mnt/churn/churn.test	churn.test	188074

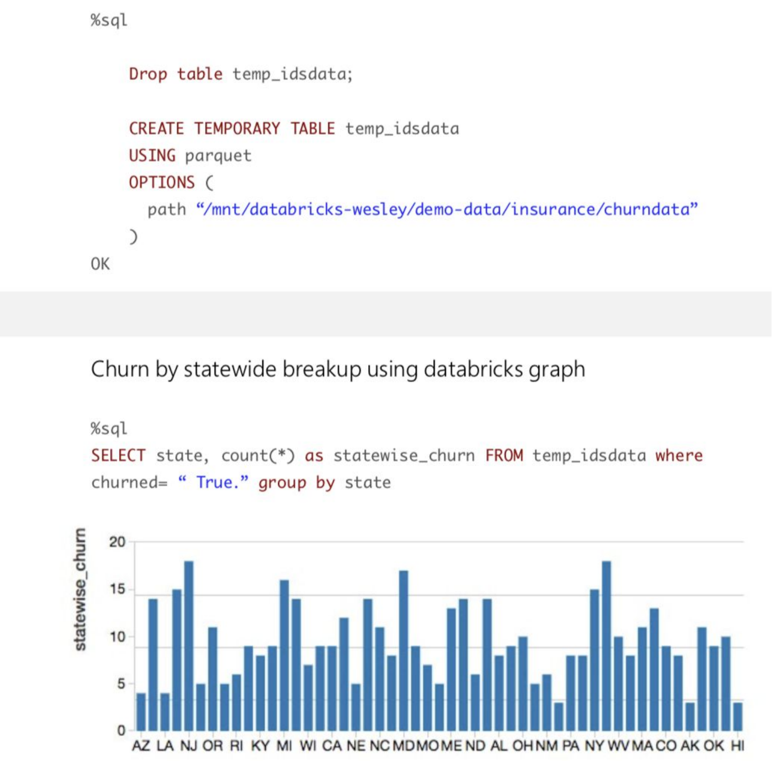
Step 4: Enrich the Data for Additional Features



```
numCases = df.count()
numChurned = df.filter(col(“churned”) == ‘ True.’).count()
```

```
numCases = numCases
numChurned = numChurned
numUnchurned = numCases - numChurned
print(“Total Number of cases: {0:,}”.format( numCases ))
print(“Total Number of cases churned: {0:,}”.format( numChurned ))
print(“Total Number of cases unchurned: {0:,}”.format( numUnchurned ))
Total Number of cases: 3,333
Total Number of cases churned: 483
Total Number of cases unchurned: 2,850
```

Step 5: Exploratory Data Analysis



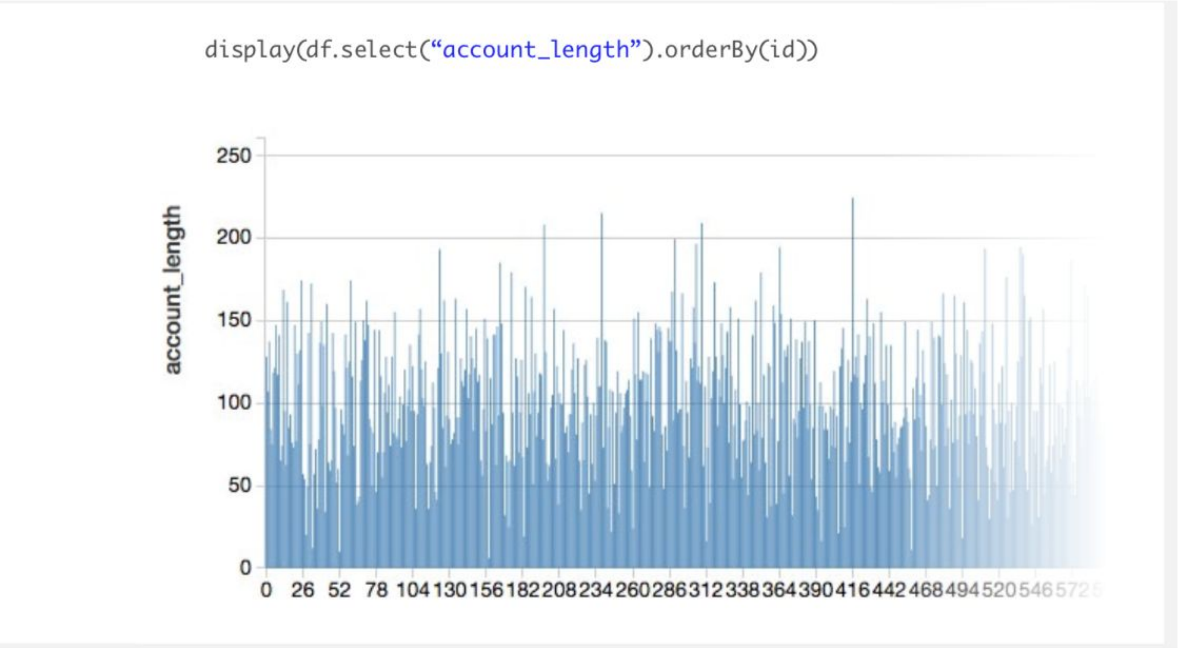
```
%sql

    Drop table temp_idsdata;

    CREATE TEMPORARY TABLE temp_idsdata
    USING parquet
    OPTIONS (
      path “/mnt/databricks-wesley/demo-data/insurance/churndata”
    )
OK
```

Churn by statewide breakup using databricks graph

```
%sql
SELECT state, count(*) as statewise_churn FROM temp_idsdata where
churned= “ True.” group by state
```

Step 6: Visualization

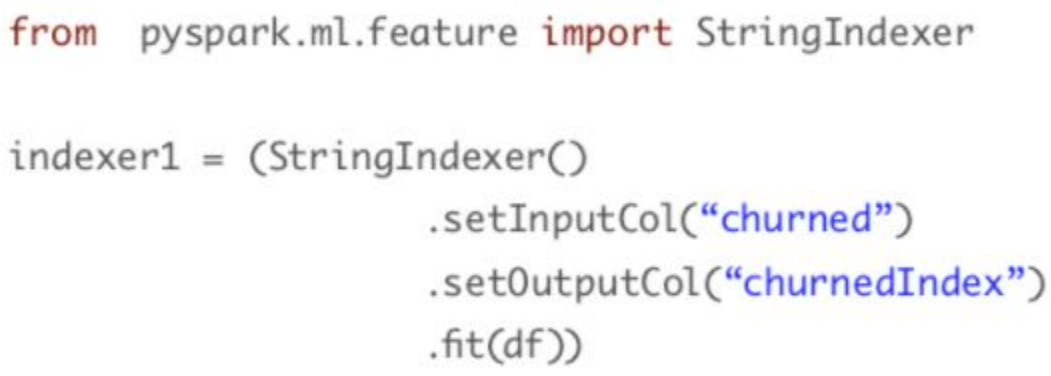
Display Length of Time with Company



```
display(df.select("account_length").orderBy(id))
```

Step 7: Model Creation (1)

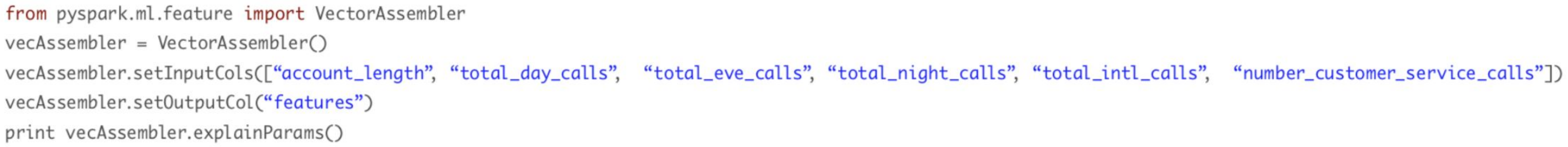
Choose/Create Target Variable: Churned, Binary Classifier



```
from  pyspark.ml.feature import StringIndexer

indexer1 = (StringIndexer()
                    .setInputCol(“churned”)
                    .setOutputCol(“churnedIndex”)
                    .fit(df))
```

Step 7: Model Creation (2)

Decide which features to keep for model creation

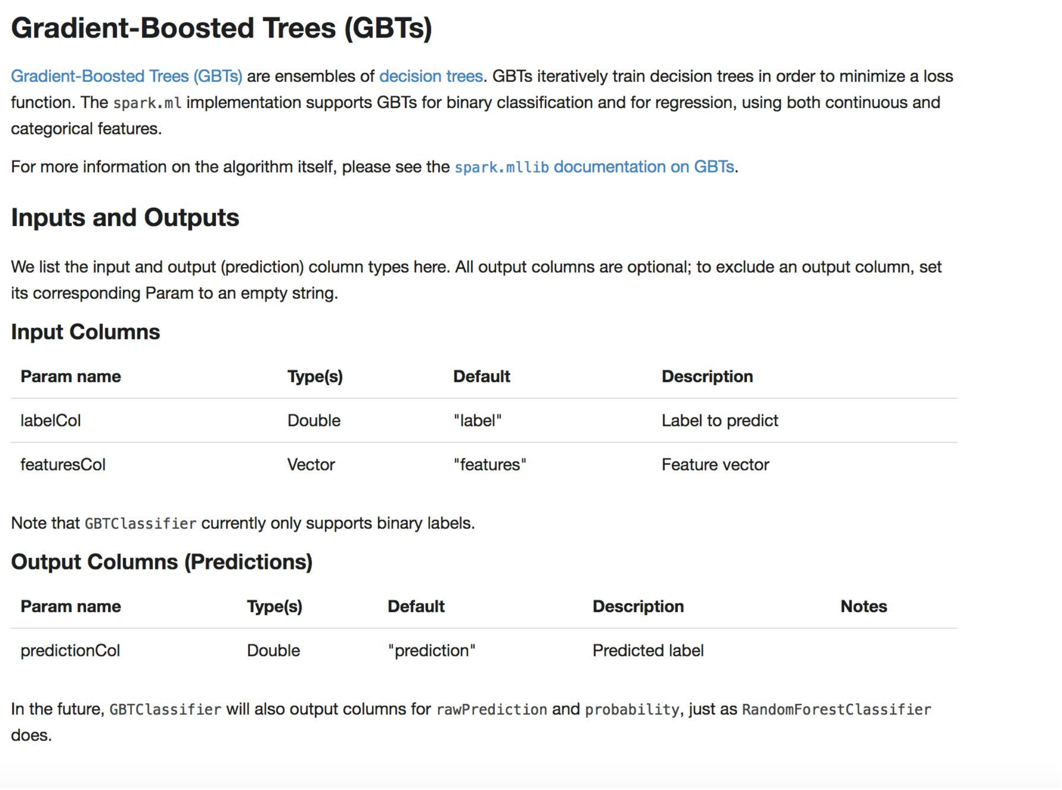


```
from pyspark.ml.feature import VectorAssembler
vecAssembler = VectorAssembler()
vecAssembler.setInputCols([“account_length”, “total_day_calls”,  “total_eve_calls”, “total_night_calls”, “total_intl_calls”,  “number_customer_service_calls”])
vecAssembler.setOutputCol(“features”)
print vecAssembler.explainParams()
```

account_length
total_day_calls
total_eve_calls
total_night_calls
total_intl_calls
number_customer_service_calls

Step 7: Model Creation (3)

Choose Machine Learning Model



Gradient-Boosted Trees (GBTs)

Gradient-Boosted Trees (GBTs) are ensembles of decision trees. GBTs iteratively train decision trees in order to minimize a loss function. The `spark.ml` implementation supports GBTs for binary classification and for regression, using both continuous and categorical features.

For more information on the algorithm itself, please see the `spark.mllib` documentation on GBTs.

Inputs and Outputs

We list the input and output (prediction) column types here. All output columns are optional; to exclude an output column, set its corresponding Param to an empty string.

Input Columns

Param name	Type(s)	Default	Description
labelCol	Double	"label"	Label to predict
featuresCol	Vector	"features"	Feature vector

Note that `GBTClassifier` currently only supports binary labels.

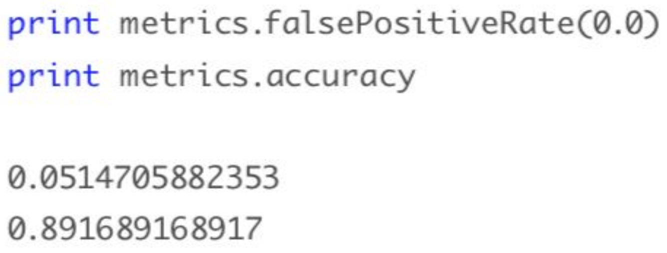
Output Columns (Predictions)

Param name	Type(s)	Default	Description	Notes
predictionCol	Double	"prediction"	Predicted label	

In the future, `GBTClassifier` will also output columns for `rawPrediction` and `probability`, just as `RandomForestClassifier` does.

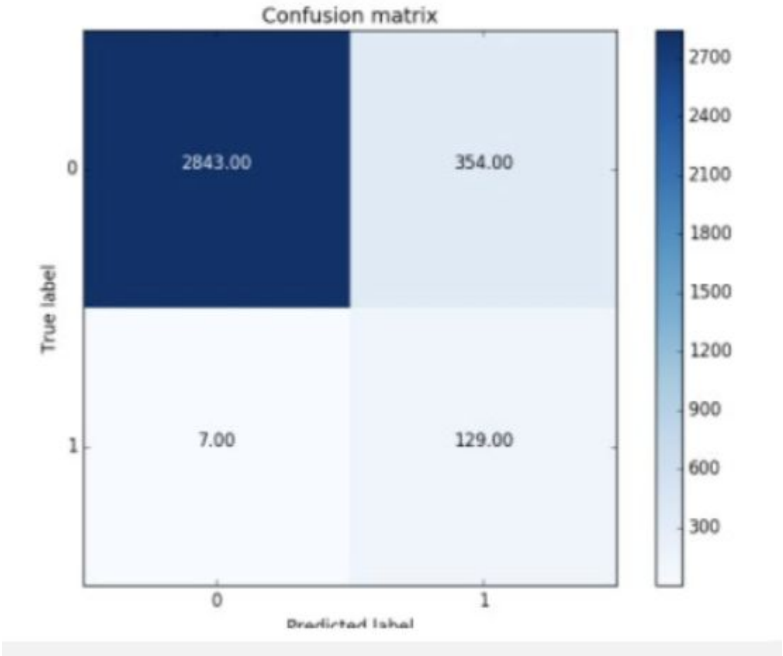
Step 8: Results Interpretation

Performance Metrics



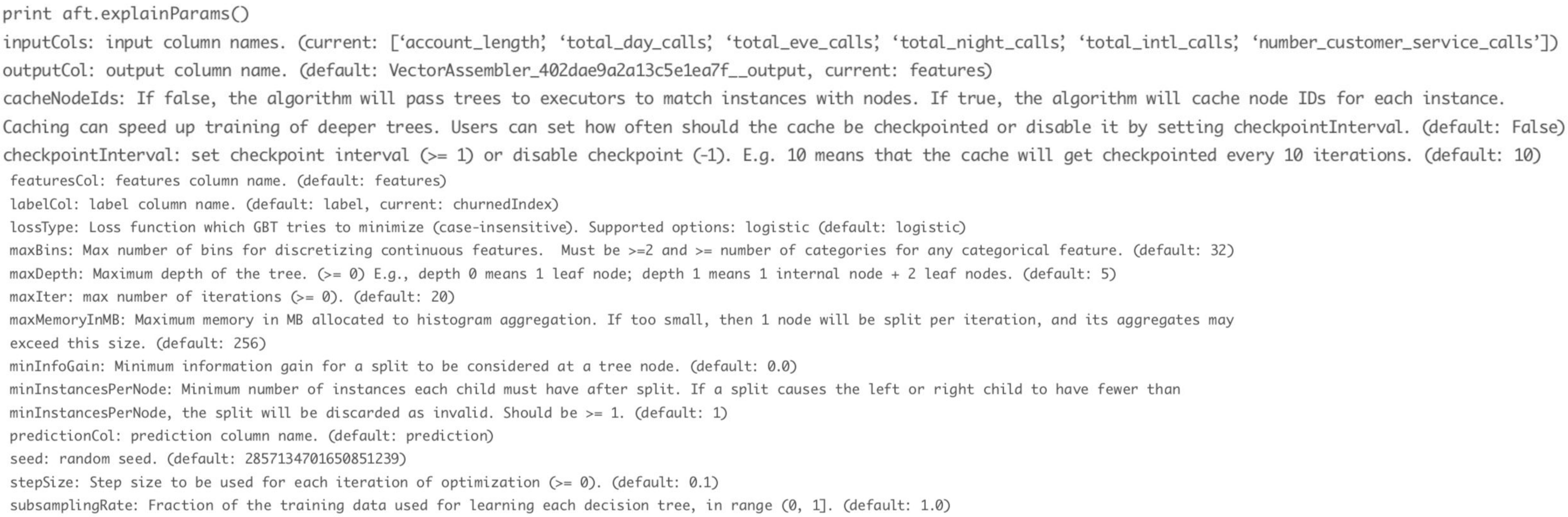
```
print metrics.falsePositiveRate(0.0)
print metrics.accuracy

0.0514705882353
0.891689168917
```

Confusion Matrix



Step 9: Next Steps

Lather, Rinse, Repeat: Tuning Parameters



```
print aft.explainParams()
inputCols: input column names. (current: [‘account_length’, ‘total_day_calls’, ‘total_eve_calls’, ‘total_night_calls’, ‘total_intl_calls’, ‘number_customer_service_calls’])
outputCol: output column name. (default: VectorAssembler_402dae9a2a13c5e1ea7f__output, current: features)
cacheNodeIds: If false, the algorithm will pass trees to executors to match instances with nodes. If true, the algorithm will cache node IDs for each instance.
Caching can speed up training of deeper trees. Users can set how often should the cache be checkpointed or disable it by setting checkpointInterval. (default: False)
checkpointInterval: set checkpoint interval (>= 1) or disable checkpoint (-1). E.g. 10 means that the cache will get checkpointed every 10 iterations. (default: 10)
 featuresCol: features column name. (default: features)
 labelCol: label column name. (default: label, current: churnedIndex)
 lossType: Loss function which GBT tries to minimize (case-insensitive). Supported options: logistic (default: logistic)
 maxBins: Max number of bins for discretizing continuous features.  Must be >=2 and >= number of categories for any categorical feature. (default: 32)
 maxDepth: Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes. (default: 5)
 maxIter: max number of iterations (>= 0). (default: 20)
 maxMemoryInMB: Maximum memory in MB allocated to histogram aggregation. If too small, then 1 node will be split per iteration, and its aggregates may
 exceed this size. (default: 256)
 minInfoGain: Minimum information gain for a split to be considered at a tree node. (default: 0.0)
 minInstancesPerNode: Minimum number of instances each child must have after split. If a split causes the left or right child to have fewer than
 minInstancesPerNode, the split will be discarded as invalid. Should be >= 1. (default: 1)
 predictionCol: prediction column name. (default: prediction)
 seed: random seed. (default: 2857134701650851239)
 stepSize: Step size to be used for each iteration of optimization (>= 0). (default: 0.1)
 subsamplingRate: Fraction of the training data used for learning each decision tree, in range (0, 1]. (default: 1.0)
```

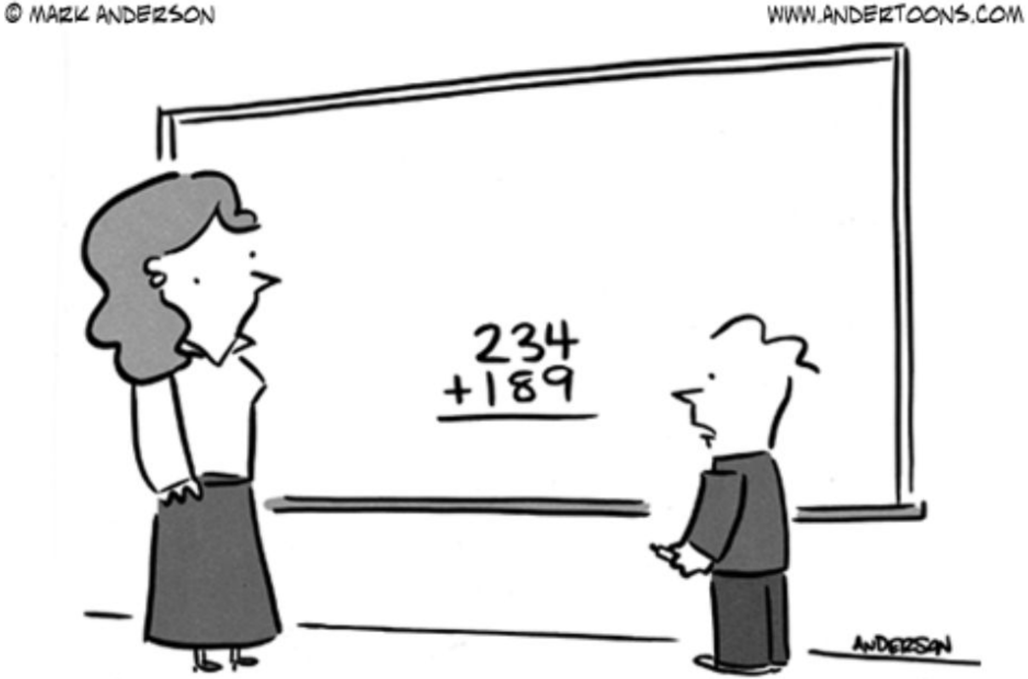
Step 10: What Now

Using predictions to generate actionable insights

Labeled vs unlabeled

Crafting the Data Story

Big Data: What is it?



“Does this count as big data?”

Resources for Hands-On Learning about Big Data

Udemy Hadoop Course:
https://www.udemy.com/the-ultimate-hands-on-hadoop-tame-your-big-data/

https://www.ngdata.com/big-data-analysis-resources/

Questions