Unsupervised Data Quality Monitoring

THREE APPROACHES TO DETERMINE BATCH ACCEPTABILITY

Sheetal Laad, Micaela Flores, Ziv Schwartz, Kelly Sooch
Data Engineering for Machine Learning
NYU Center For Data Science

Introduction & Background

Task

Given a chronological series dataset batches, decide whether the next batch has acceptable data quality

Goal

Create and compare different methodologies that can discern which data batches are sufficiently clean by only training the model on a sample of clean batches

Motivation

When working with large amounts of data, it becomes too difficult to manually check if each input data batch maintains its integrity. Providing a dynamic data auditing system that performs unsupervised data quality monitoring given a chronological set of clean and dirty data batches to combat this issue.

Data

Data is comprised of Flights data and Facebook Posts

Facebook: 53 clean csv files and 53 dirty csv files

- 'line', 'page',
- 'week', 'num_likes',
- 'domain', 'outlet',
- 'title', 'description',
- 'contenttype', 'image',
- 'url', 'text',
- 'id', 'right_of_center'

Flights: 31 clean csv files and 31 dirty csv files

- 'RowId', 'Source',
- 'Flight', 'ScheduledDeparture',
- 'ActualDeparture', 'DepartureGate'
- 'ScheduledArrival', 'ActualArrival',
- 'ArrivalGate', 'for_key', 'date'

Data

Example Flight Data (same batch)

Clean

RowId	Source	Flight	ScheduledDeparture	ActualDeparture	DepartureGate	ScheduledArrival	ActualArrival	ArrivalGate	for_key	date
0	ua	UA-2708-EWR-CLT	Thu_ Dec 1 2:55 PM	Thu_ Dec 1 2:55 PM	A37	Thu_ Dec 1 4:53 PM	Thu_ Dec 1 4:44 PM	C5	Thu_ Dec 1 2:55 PM	734472
1	airtravelcenter	UA-2708-EWR-CLT	Thu_ Dec 1 2:55 PM	Thu_ Dec 1 2:55 PM	A37	Thu_ Dec 1 4:53 PM	Thu_ Dec 1 4:44 PM	C5	Thu_ Dec 1 2:55 PM	734472
2	myrateplan	UA-2708-EWR-CLT	Thu_ Dec 1 2:55 PM	Thu_ Dec 1 2:55 PM	A37	Thu_ Dec 1 4:53 PM	Thu_ Dec 1 4:44 PM	C5	Thu_ Dec 1 2:55 PM	734472
3	helloflight	UA-2708-EWR-CLT	Thu_ Dec 1 2:55 PM	Thu_ Dec 1 2:55 PM	A37	Thu_ Dec 1 4:53 PM	Thu_ Dec 1 4:44 PM	C5	Thu_ Dec 1 2:55 PM	734472
4	flytecomm	UA-2708-EWR-CLT	Thu_ Dec 1 2:55 PM	Thu_ Dec 1 2:55 PM	A37	Thu_ Dec 1 4:53 PM	Thu_ Dec 1 4:44 PM	C5	Thu_ Dec 1 2:55 PM	734472

Dirty

RowId	Source	Flight	ScheduledDeparture	ActualDeparture	DepartureGate	ScheduledArrival	ActualArrival	ArrivalGate	date
0	ua	UA-2708-EWR-CLT	Thu_ Dec 1 2:55 PM	Thu_ Dec 1 2:55 PM	A37	Thu_ Dec 1 4:53 PM	Thu_ Dec 1 4:44 PM	C5	734472
1	airtravelcenter	UA-2708-EWR-CLT	NaN	12/1/11 3:04 PM (-05:00)	NaN	NaN	12/1/11 4:22 PM (-05:00)	NaN	734472
2	myrateplan	UA-2708-EWR-CLT	NaN	12/1/11 3:04 PM (-05:00)	NaN	NaN	12/1/11 4:22 PM (-05:00)	NaN	734472
3	helloflight	UA-2708-EWR-CLT	NaN	12/1/11 3:04 PM (-05:00)	NaN	NaN	12/1/11 4:22 PM (-05:00)	NaN	734472
4	flytecomm	UA-2708-EWR-CLT	NaN	12/1/11 3:04 PM (-05:00)	NaN	NaN	12/1/11 4:22 PM (-05:00)	NaN	734472

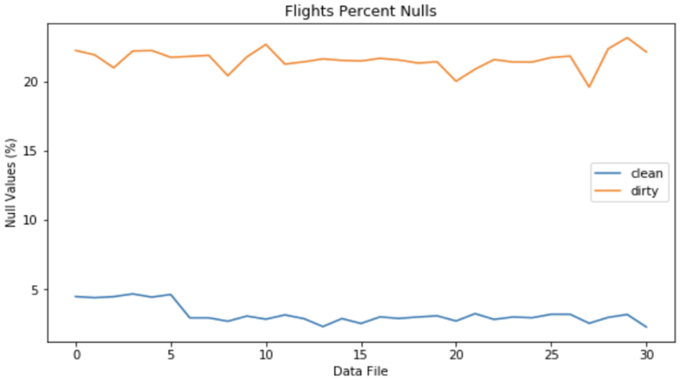
Training Methodology

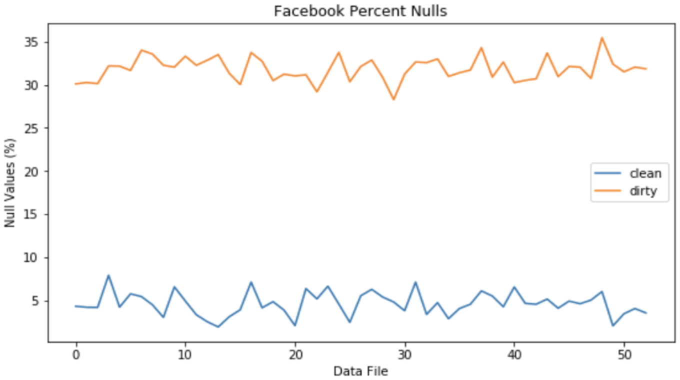
There were two different ways in which our group decided to handle the batch size used to train our models:

- **Increasing**:
  - Predetermined starting batch size is set and for every new batch that is read in, it is included as part of the whole training set.
  - Example: Starting batch size of 3, the first example would get trained on clean batches 1 to 3 and batch 4 will be tested. For the second example, clean batches 1 to 4 would be used to train and batch 5 will be tested.
- **Rolling**:
  - Predetermined batch size is used for the training criterion and moves from subsequent batch to batch
  - Example: Batch size set to 3, first example trained on batches 1 to 3 and batch 4 is tested. Second example is trained on batches 2 to 4, batch 5 tested.

Criterion Approach 1: Baseline Methodology

Percentage of null values per data batch





Criterion Explained:

- Percentage of null values in each batch of the training data
- If the percentage of nulls in the test batch is not within the minimum and maximum percentage of nulls of the training batches, the batch is classified as not acceptable

Criterion Approach 2: Range Methodology

- Uniqueness = count of distinct values by field
- Completeness = count of records that have a missing value for that field / count of records that don't have a missing value by field

Aggregated the mean values across all batches for each given column:

Uniqueness

Clean	
RowId	2381.290323
Source	37.612903
Flight	93.483871
ScheduledDeparture	82.935484
ActualDeparture	86.516129
DepartureGate	67.000000
ScheduledArrival	90.354839
ActualArrival	86.322581
ArrivalGate	66.838710
for_key	82.935484
date	1.000000

Dirty	
RowId	2381.290323
Source	37.612903
Flight	93.483871
ScheduledDeparture	508.064516
ActualDeparture	662.903226
DepartureGate	147.225806
ScheduledArrival	588.225806
ActualArrival	728.419355
ArrivalGate	146.225806
date	1.000000

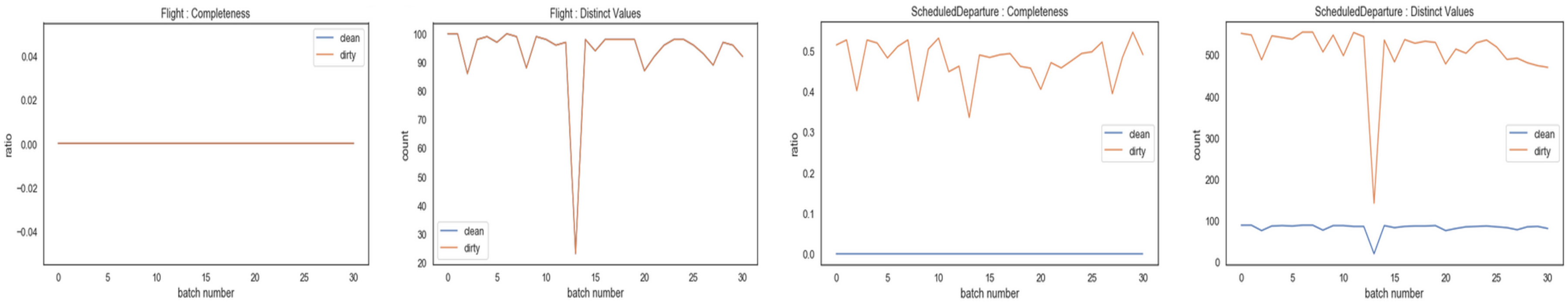
Completeness

Clean	
RowId	0.000000
Source	0.000000
Flight	0.000000
ScheduledDeparture	0.000000
ActualDeparture	0.014968
DepartureGate	0.188292
ScheduledArrival	0.000000
ActualArrival	0.025478
ArrivalGate	0.188728
date	0.000000

Dirty	
RowId	0.000000
Source	0.000000
Flight	0.000000
ScheduledDeparture	0.476794
ActualDeparture	0.190283
DepartureGate	1.503791
ScheduledArrival	0.464570
ActualArrival	0.197457
ArrivalGate	1.488697
date	0.000000

Criterion Approach 2: Range Methodology

Completeness and Uniqueness by batch



Criterion Explained:

- If 80% of fields are within +/- 2 stds of completeness mean and of uniqueness mean, batch is determined to be acceptable

Criterion Approach 3: TensorFlow Data Validation

Example of anomalies for a field:

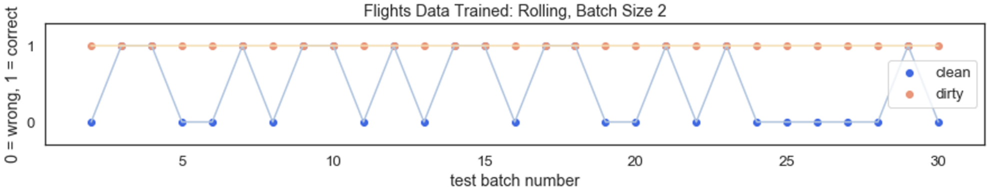
Feature name	Anomaly short description	Anomaly long description
'DepartureGate'	Unexpected string values	Examples contain values missing from the schema: - (~2%), 134 (~1%), 15? (<1%), 27/T2 (<1%), 29 (<1%), 3/8 (<1%), 33/8 (<1%), 37 (<1%), 40/8 (<1%), 41 (<1%), 41/8 (<1%), 44E (<1%), 44F/T4 (<1%), 45/T4 (<1%), 48B (<1%), 48B/T4 (<1%), 6/3 (<1%), 60 (<1%), 62 (<1%), 64 (<1%), 7 (<1%), 70A (<1%), 70B (~1%), 8/8 (<1%), 88 (<1%), 92 (<1%), A1 (<1%), A14 (<1%), A14$ (<1%), A7/A (<1%), B22 (<1%), B3 (<1%), B8/B (<1%), C-33 (<1%), C11 (~3%), C134 (<1%), C16 (~1%), C16/C (<1%), C19 (<1%), C19/C (<1%), C30/C (<1%), C40 (<1%), C41 (<1%), C8/C (<1%), C88 (<1%), CHK (<1%), D20/D (<1%), D28 (~1%), D28/D (<1%), D30 (~1%), D32 (<1%), D4 (<1%), D48 (<1%), D75 (<1%), E1 (<1%), E2 (<1%), E8 (<1%), G14 (<1%), G19A/T3 (<1%), G1B/T3 (<1%), G3/T3 (<1%), G9/T3 (<1%), G92 (<1%), H15/T3 (<1%), K8/T3 (<1%), K9/T3 (<1%), Main Term - C25 (<1%), Not provided by airline (~1%), Term C - C134 (<1%), Term C - C33 (<1%), Term C - E1 (<1%), Terminal 2 (~1%), Terminal 2 - 57 (<1%), Terminal 3 (~1%), Terminal 3 - E3 (<1%), Terminal 4 (<1%), Terminal 4 - 44F (<1%), Terminal 4 - 45 (<1%), Terminal 4 - 48B (<1%), Terminal 6 (<1%), Terminal 8 - 3 (<1%), Terminal 8 - 33 (<1%), Terminal 8 - 40 (<1%), Terminal 8 - 41 (<1%), Terminal 8 - 8 (<1%), Terminal A - 15 (<1%), Terminal D - D1 (<1%), Terminal D - D36 (<1%), Terminal D - D40 (<1%), Terminal D - D48 (<1%).

Criterion Explained:

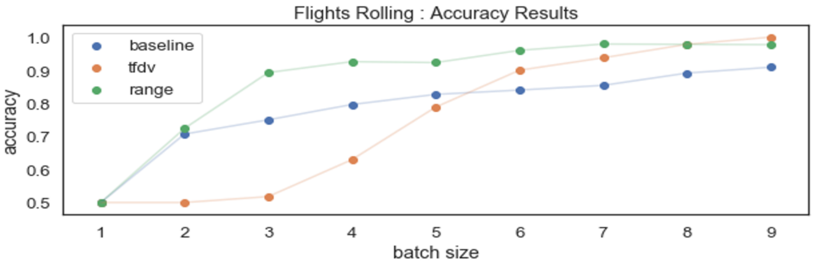
- TensorFlow Data Validation library to create an acceptable schema from the training batches and measure the number anomalies
- Anomalies are counted before and after schema adjustment, which consists of relaxing certain columns' domain mass requirements in the trained schema
  - Flights: 'DepartureGate' and 'ArrivalGate' were both reduced to 80% adherence
  - Facebook: 'outlet' and 'domain' were relaxed to 65% and 60% adherence, respectively
- Then number of anomalies is calculated again, using the same method from before. If the schema adjustment reduces the number of anomalies, then the test batch is considered acceptable

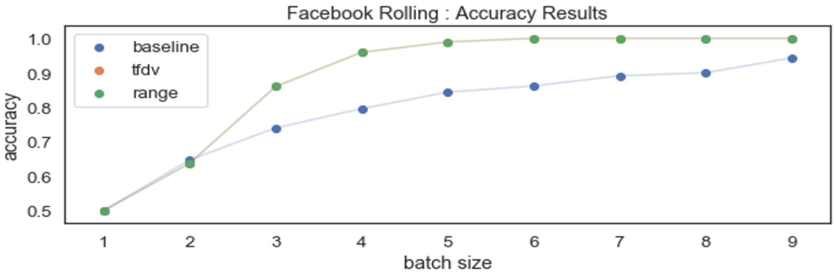
Evaluation/Results

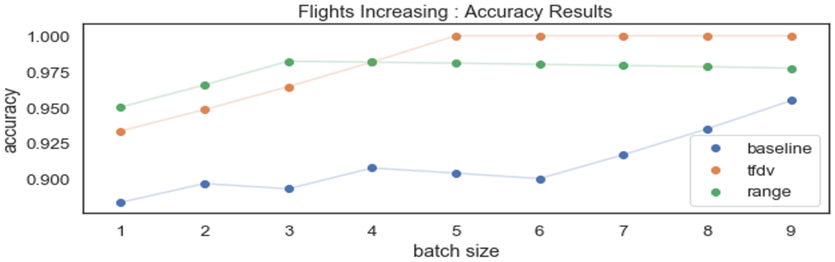
Example of experiment results:

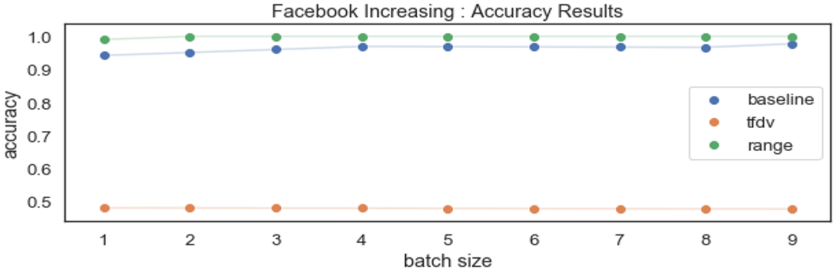


Final Results:









Discussion

The results of this project provides great insight into data quality monitoring in a production environment.

- There exists an optimal batch size
- Rolling training methods: faster runtimes, give less importance on older information (helpful in production when a field starts getting recorded different, when data seems to be changing, to capture seasonal trends)
- Range method: faster runtimes, more intuitive and generalizable

Some of the challenges that we faced:

- Inherent differences in production batches (Flights data)
- *TensorFlow Data Validation*’s very long runtime
- *TensorFlow Data Validation* is not very generalizable

Future work for this topic could include:

- Looking more at the data values themselves
- Involving more analysis of data trends (seasonal, yearly, etc.)